VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY PRO DOPLŇOVÁNÍ PIXELŮ VNĚ OBRAZU

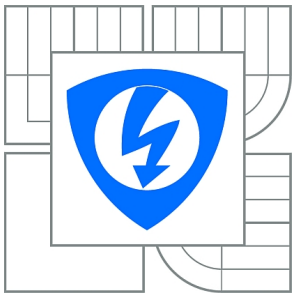
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR JEŠKO
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

METODY PRO DOPLŇOVÁNÍ PIXELŮ VNĚ OBRAZU

IMAGE EXTRAPOLATION METHODS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

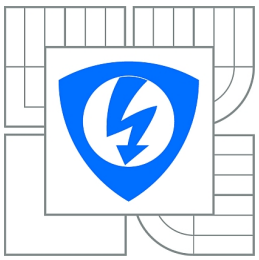
AUTOR PRÁCE — Bc. PETR JEŠKO

AUTHOR

VEDOUCÍ PRÁCE — Mgr. PAVEL RAJMIC, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Petr Ješko ***ID:*** 119467
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Metody pro doplňování pixelů vně obrazu

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte z dostupné literatury různé přístupy k doplňování chybějících oblastí v obrazu (angl. tzv. inpainting) - výpadek přenosového kanálu, ostraněné logo, škrábanec či kaňka na fotografii. Inpsirujte se těmito přístupy, navrhněte a implementujte několik algoritmů, které doplňují pixely "za okrajem" obrazu. Ověřte a srovnejte na testovacích obrázcích rozličného charakteru.

DOPORUČENÁ LITERATURA:

[1] G. Aubert and P. Kornprobst, Mathematical Problems in Image Processing. Partial Differential Equations and the Calculus of Variations, Springer-Verlag, New York, 2002.
[2] Masnou, S. Disocclusion: a variational approach using level lines, Image Processing, IEEE Transactions on , vol.11, no.2, pp.68-76, Feb 2002, doi: 10.1109/83.982815
[3] C. Ballester et al. Filling-in by joint interpolation of vector fields and gray levels, IEEE Trans. Image Process., 10 (2001), pp. 1200–1211.
[4] M. Beltramio, G. Sapiro, V. Caselles, and B. Ballester, Image inpainting, in Proceedings of the 27th Annual ACM Conference on Computer Graphics, 2000, pp. 417–424.
[5] MISITI, Michel, et al. Wavelet Toolbox : For Use with MATLAB. The MathWorks Inc., 2000.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Mgr. Pavel Rajmic, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ABSTRAKT

Práce se zabývá problematikou doplňování pixelů vně obrazu. Uvádí některé metody pro inpainting použitím počítače a upozorňuje na úskalí, která se zde objevují. Zkoumá metody pro interpolaci a aproximaci funkcí ve snaze najít nejlepší metodu pro extrapolování obrazu za jeho hranice. Popisuje základy Waveletové transformace a Multiresolution analýzy. Navrhuje několik metod pro doplňování pixelů vně obrazu. Porovnává dosažené výsledky pomocí PSNR a SSIM. Tyto metody jsou zde vysvětleny a srovnány. Stručně pojednává o algoritmu OMP, spadajícím do oblasti řídké reprezentace signálů, a použitém v jedné z metod. Také je zde představeno vývojové prostředí MATLAB jako nástroj pro implementaci algoritmů, které prakticky řeší zadanou problematiku. V praktické části jsou popsány implementované metody pro doplňování pixelů vně obrazu.

KLÍČOVÁ SLOVA

aproximace, extrapolace, inpainting, interpolace, MATLAB, MR analýza, OMP, PSNR, SSIM, Waveletová transformace

ABSTRACT

The thesis deals with addition of pixels outside the image. Lists some methods for inpainting using computers and highlights the pitfalls that appear here. Examines methods for interpolation and approximation of functions in order to find the best method for extrapolating the image beyond its borders. Describes the basics of Wavelet transformation and Multiresolution analysis. It is proposed several methods for replenishment of pixels outside the image. PSNR and SSIM are used to compare achieved results. These methods are explained and compared. Briefly discusses the algorithm OMP, falling within the sparse representation of signals, and used in one of the methods. Also discussed is the development environment of MATLAB as a tool for the implementation of algorithms that practically solves the given problem. The practical part describes the implemented methods for adding pixels outside the image.

KEYWORDS

approximation, extrapolation, inpainting, interpolation, MATLAB, MR analysis, OMP, PSNR, SSIM, Wavelet transform

JEŠKO, Petr *Metody pro doplňování pixelů vně obrazu*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 76 s. Vedoucí práce byl Mgr. Pavel Rajmic, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Metody pro doplňování pixelů vně obrazu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Mgr. Pavlu Rajmicovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Také bych rád poděkoval Ing. Aleně Vašatové za přínosné konzultace možností systému MATLAB.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Inpainting je proces rekonstrukce ztracených nebo zničených částí obrazu nebo videa. V případě potřeby obnovit hodnotnou malbu se procesu rekonstrukce ujme zkušený umělec. V digitálním světě se termín inpainting odvolává na sadu různých sofistikovaných algoritmů, které zajistí opravu ztracených nebo poškozených obrazových dat. Tyto algoritmy se rovněž dají uplatnit při potřebě korekce po nějaké geometrické transformaci. Jako příklad může posloužit potřeba doplnění pixelů na okrajích fotografie po korekci tzv. soudkovitosti fotografie nebo vyplňování černých oblastí v panoramatických fotografiích.

V posledních dvou případech a v některých dalších požadujeme prodloužení obrazu do jednoho, či více směrů. Tzn. doplnění pixelů za hranici obrazu. A právě k tomu slouží speciální případ inpaintingu, kterým se zabývá tato práce. [1]

Nyní si pojďme říct, jak vlastně inpainting funguje. Počítač nevidí obraz tak komplexně jako člověk, proto je třeba mu definovat oblasti, kde se nacházejí defekty nebo objekty, které chceme odstranit nebo vylepšit. Jedna z možností je označit tyto oblasti ručně. V případě, že chceme vylepšit celý obraz je pro uživatele nejvíce výhodné, když si umí počítač najít oblasti poškození sám. Metodě, která k tomu slouží se říká detekce.

Potom nastupuje samotný inpainting. Ten řeší problém vyplňování chybějících oblastí (text přes fotografii, ohyb, škrábanec). Nebo překrytí objektu jiným objektem (turista stojící před jezerem nám kazí kompozici fotografie jezera). Zjednodušeně lze říci, že na základě hodnot okolních pixelů opravované oblasti vypočítá algoritmus předpokládaný průběh okolí až do plochy, kde se nachází opravovaná oblast. Pro tento úkon existuje řada možných způsobů řešení, o kterých si povíme později. Je zde ale důležité poznamenat, že inpainting jako takový se používá k vyplňování oblastí uvnitř obrazu, kde můžeme pro výpočet použít informace o hodnotě pixelů z okolí ze všech směrů. Tato práce se však zabývá doplňováním pixelů za hranicí obrazu, z čehož plyne možnost použití hodnot okolních pixelů jen z jednoho směru.

Třetím krokem může být restaurování. To zajistí např. vyhlazení přechodů. V některých případech to však není žádoucí. Obzvláště, když potřebujeme zachovat detaily.

Jak už jste měli šanci vyvodit z předchozího textu, aplikací pro inpainting je opravdu hodně. Principiálně jsou si všechny podobné, ale pro představu uvedu krátký výčet možných aplikací. Odstraňování osob, vodoznaků, elektrického vedení, razítek s datem nebo informačních tabulí z fotografií nebo videa. Redukce červených

[1] Inpainting je umělecké synonymum pro image interpolation, tzn. interpolaci obrazu. Jak uvádí [1, Shen]

Obr. 1: Stará popraskaná malba (vlevo) a tatáž malba po odstranění popraskání pomocí inpaintingu (vpravo). Obrazy pocházejí z www.brothersoft.com

očí. Odstraňování šumu. Klonování objektů na fotografii. Syntéza textury. Nahrazování ztracených bloků při kódování a přenosu obrazu nebo videa. Vyplňování černých oblastí v panoramatických fotografiích nebo úprava fotografií po geometrické transformaci.

1 INPAINTING

1.1 Techniky inpaintingu

Jak již bylo nastíněno v úvodu, inpainting se využívá k široké škále různých úloh. V závislosti na použití se tudíž počítačové zpracování opírá o různé techniky a přístupy. Od plně automatických metod, které samy vyhledávají poškození, po metody, kde si uživatel sám přesně nadefinuje, kterou oblast je potřeba opravit. Oba přístupy mají své výhody i nevýhody. První nevyžaduje přílišnou spolupráci uživatele a šetří tak čas, naproti tomu výsledky zde nebývají tak kvalitní jako u druhého přístupu. Nevýhody prvního jsou zároveň výhodami druhého a naopak. Účelem této práce není řešit inpainting uvnitř obrazu. Znalosti v tomto oboru jsou pro náš případ doplňování pixelů vně obrazu ale přínosné. Představíme zde tedy alespoň některé jednodušší metody, aby čtenář získal rámcovou představu o problému. Nebude se jednat o podrobný popis výpočetních algoritmů, spíše o popis principů řešení. Následující text je převzat z [2] a zabývá se případem vymazání objektu z obrazu a následným vyplněním plochy pozadím. A to Exemplar based metodou (Exemplární), metodou Poissonových rovnic, metodou postupné eliminace a metodou osmi sousedních pixelů.

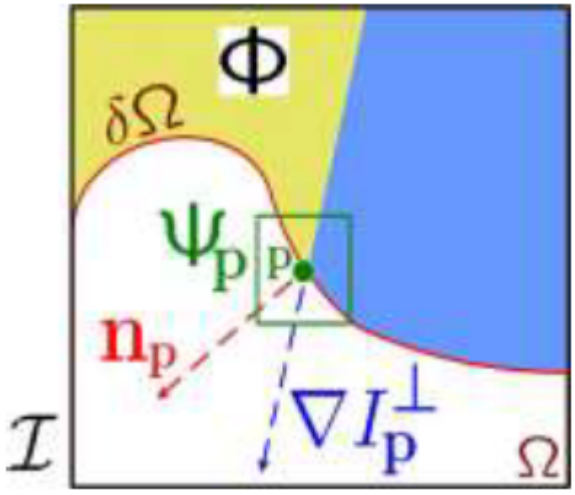
1.1.1 Exemplar-based region filling

Název metody vystihuje její podstatu. Jedná se o vyplňování oblastí obrazu nejvíce podobným blokem nalezeným v jiné části obrazu. Uživatel vybere cílovou oblast Ω, která bude vymazána a následně vyplněna. Zdrojová oblast Φ, může být definována jako celý obrázek mínus cílový region ($\Phi = \mathrm{I} - \Omega$), rozšířené pásmo okolo cílové oblasti nebo může být ručně specifikováno uživatelem. Algoritmus opakuje následující tři kroky do té doby, než jsou všechny pixely vyplněny.

- vypočítá priority místa
- syntézu textury
- pořadí plnění

Počítání priorit místa

Algoritmus vykonává syntézu prostřednictvím strategie „nejlepší napřed“, která zcela závisí na hodnotách priority, které jsou přiřazeny každému místu na hranici doplňovaného prostoru. Výpočet priority je ovlivněn místy, které (a)jsou pokračováním výrazných hran a (b)jsou obklopeny pixely s vysokou mírou jistoty. Danému místu p definujeme jeho prioritu $P(p)$ jako násobek dvou podmínek.



Obr. 1.1: Diagram notace. Zdroj obrazu je [2].

$$\mathrm{P(p)} = \mathrm{C(p)} * \mathrm{D(p)}, \tag{1.1}$$

kde C(p) je podmínka jistoty a D(p) podmínka údajů (fakt),

$$\mathrm{C(p)} = (\Sigma q \in \Psi p \cap (I - \Omega)c(q))/|\Psi p|, \tag{1.2}$$

$$\mathrm{D(p)} = |(\nabla I \perp p.np)|/\alpha. \tag{1.3}$$

Podmínka jistoty $C(p)$ může být chápána jako ukazatel množství spolehlivé informace obklopující pixel p. Záměrem je vyplnit nejprve ta místa, která mají již nějaké pixely v okolí vyplněny, s dodatečnou prioritou danou pixelům, které byly vyplněny dříve nebo nikdy nebyly částí cílové oblasti.

Místa, která obsahují rohy a tenké úponky cílové oblasti budou vyplněna první, protože jsou obklopena více pixely z originální části obrazu. Tato místa poskytují více spolehlivé informace pro hledání shody. Naopak místa na špičce „poloostrovů“ budou odložena stranou do té doby, než budou vyplněny hodnoty více obklopujících pixelů.

Syntéza textury

Jakmile jsou vypočteny všechny priority pro pixely na hranici cílové oblasti, je vypočítáno místo p s nejvyšší prioritou. To potom naplníme daty extrahovanými ze zdrojové oblasti. Ve tradičním přístupu k inpaintingu je informace o hodnotě pixelu propagována skrze difuzi. Jak bylo poznamenáno výše, difuze nutně vede k vyhlazování obrazu, důsledkem čehož se takto vyplněná plocha jeví rozmazaně. A to především při vyplňování velkých oblastí. Přístup této metody naopak je texturování

obrazu přímým vzorkováním zdrojové oblasti. Ve zdrojové oblasti se hledají místa, která jsou nejvíce podobná tomu právě vyplňovanému. Formálně

$$\Psi q = \arg\min d(\Psi p, \Psi q), \tag{1.4}$$

$$\Psi q \in \Phi. \tag{1.5}$$

Pořadí vyplňování

Výše popsaná metoda může být schopna propagovat texturní i strukturní informace. Tento odstavec vysvětluje, že kvalita syntézy výstupního obrazu je velmi ovlivněna pořadím, v jakém se vyplňovací proces provádí. Řazení vyplněných míst vytváří horizontální hranici mezi oblastmi pozadí obrázku.

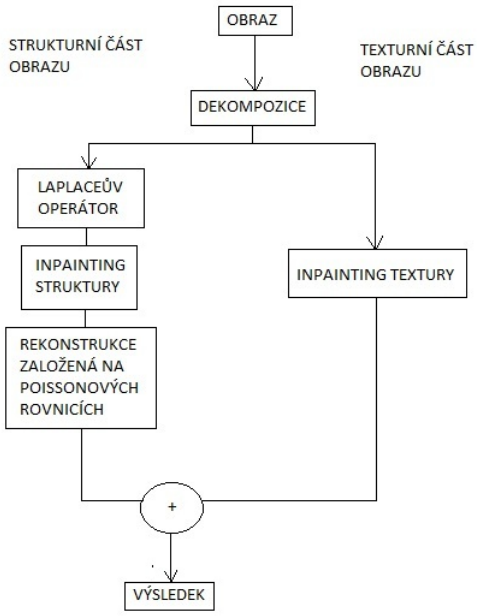
Řazení soustředné vrstvy v kombinaci s vyplňováním na bázi místa může produkovat další artefakty. Další žádoucí vlastností dobrého vyplňovacího algoritmu je zamezení tzv. „over-shootingu“ artefaktů. Volně přeloženo nadměrnému vytváření artefaktů. Což se stává, pokud je hranám (okrajům) obrazu povolen nekonečný růst.

Algoritmus pro vyplňování oblastí

1. Najít všechny hraniční body přerušené (roztržené) části.
2. Najít šablonu a jistotu pro každý hraniční bod.
3. Znásobit datovou šablonu a jistotu každého hraničního bodu (tj. celkový bod/ vliv/prioritu jednotlivého pixelu).
4. Najít hraniční bod s nejvyšší prioritou.
5. Vytvořit okno (záplatu) s pixelem s nejvyšší prioritou uprostřed.
6. Najít shodu (stejný vzor/okno) v celém obrazu od souřadnice (0,0) do (x,y) a identifikovat toto okno.
7. Shromáždit pozice všech shodných oken a opět najít datovou šablonu a jistotu pro každé nalezené okno. Vynásobit datovou šablonu a jistotu.
8. Najít shodné okno s nejvyšší prioritou a potom jím nahradit vyplňované místo, které bylo vytvořeno v bodě 5
9. Opakovat dokola dřívější kroky až do doby, kdy již není potřeba vyplnit žádné místo.

1.1.2 Poissonova rovnice

Při této metodě se obraz rozloží do dvou složek. Na strukturu a texturu. Potom se opraví obě složky zvlášť podle jejich charakteristiky. Textura je opravena pomocí exemplární metody. V případě struktury je využito Laplaceova operátoru k vylepšení



Obr. 1.2: Diagram inpaintingu. Zdroj obrazu je [2].

informací o struktuře. A Laplaceovský obraz je vyplněn exemplárním algoritmem následovaným rekonstrukcí založenou na Poissonově rovnici.

Ve většině existujících metod pro inpainting je oblast inpaintingu vypočtena jako nejhladší pokračování lokální struktury, kde hladkost může být definována různými způsoby. Avšak tyto předpoklady se stávají nevhodnými, pokud se velikost oblasti pro inpainting zvětšuje nebo je uvnitř textura. Tyto metody navíc využívají pouze malý radius okolí, obklopující oblast inpaintingu, aby ji opravily. To vede k tomu, že takovéto algoritmy jsou náchylné na šum a tudíž nevhodné pro opravu rozlehlých oblastí. K tomu všemu pokud oblast, která má být opravena obsahuje obojí, tzn. texturu i strukturu, textura bude narušovat opravný proces struktury, což způsobuje chybné hrany ve zrekonstruované části obrazu.

Poissonův algoritmus

1. Najít část, která má být vykreslena.
2. Rozdělit obraz do dvou složek na texturu a strukturu.
3. Pro texturu se provede to samé, co v exemplární metodě.
4. Vyhladit strukturu pomocí Laplaceova operátoru.
5. Zkombinovat oba výstupy získané v bodě 3 a 4.
6. Výsledkem je obnovený obraz.

1.1.3 Inpainting založený na postupné eliminaci

Algoritmus postupné eliminace (SEA) je využit ke zefektivnění exemplární metody a k získání optimálního globálního řešení. Základní myšlenka je získat nejlepší odhad vektorů postupným vynecháváním pozic ve vyhledávacím okně a tedy snížit počet shodných hodnocení, které vyžaduje velmi intenzivní výpočty.

Výpočetní efektivita je zvýšena pomocí výše zmíněného SEA, který je založen na součtu absolutních rozdílů (SAD), aby získal globálně optimální řešení. Definice SAD je tato:

$$d(\Psi p, \Psi q) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f(x+i, y+j) - f(m+i, n+j)| \tag{1.6}$$

$$M(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f(m+i, n+j)| \tag{1.7}$$

$$R = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f(x+i, y+j)| \tag{1.8}$$

To znamená

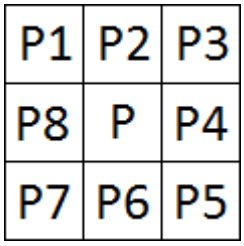
$$R - SAD(u,v) \leq M(m,n) \leq R + SAD(u,v). \tag{1.9}$$

Algoritmus postupné eliminace

1. Najít všechny hraniční body opravované oblasti.
2. Najít šablonu dat a jistotu pro každý hraniční bod.
3. Znásobit datovou šablonu a jistotu každého hraničního bodu.
4. Najít hraniční pixel s největší prioritou.
5. Vytvořit okno (záplatu) s pixelem s nejvyšší prioritou uprostřed.
6. Najít shodu (stejný vzor/okno) v celém obrazu od souřadnice (0,0) do (x,y) a identifikovat toto okno.
7. Zastavit proces hledání další shody, jakmile je jedna nalezena (U exemplární metody bychom hledali všechny možné shody, potom z nich shodu s nejvyšší prioritou a nakonec vybrali nejlepší volbu. V případě použití SEA ale další shody již nedosáhneme.)
8. Nahrazení okna vytvořeného v bodě 5 právě nalezeným oknem.
9. Opakovat dokola dřívější kroky až do doby, kdy již není potřeba vyplnit žádné místo.

1.1.4 Metoda rychlého rozmazání okolních osmi pixelů

Okolí pixelu P je množina okolních osmi pixelů, které sdílejí hranu nebo vrchol s daným pixelem P. Jmenovitě jsou to P1, P2, P3, P4, P5, P6, P7 a P8, jak ukazuje obrázek.



Obr. 1.3: 8 sousedních pixelů k pixelu P.

Metody popsané výše dosahují dobrých výsledků na určitých typech obrazu, ale jsou zde zřejmá omezení a nelze je tedy použít pro každý případ potřeby. Většina z nich je založena na iterativním přístupu a jeden bod vyžaduje pro obnovu často tisíců iterací. To zabere samozřejmě spoustu času a obnova jednoho obrazu tak může nezřídka zabrat i několik minut, což omezuje propagaci a aplikaci těchto algoritmů. Kvůli této situaci byla vymyšlena metoda rychlého rozmazání okolních osmi pixelů. Ta se snaží zachovat efektivitu obnovování a zkrátit výrazně čas zpracování.

Protože jednotlivé metody inpaintingu včetně této jsou poměrně složité a rozsáhlé, nebudeme se touto problematikou hlouběji zabývat. Pro případný zájem ale uvádíme zdroj, ve kterém je tato metoda popsána [3]. Je však vhodné uvést alespoň orientační srovnání časových nároků na výpočty pomocí zmíněných metod, aby měl čtenář rámcovou představu.

Tab. 1.1: Přehled potřebného výpočetního času pro popsané metody

Typ obrazu	Exemplární metoda	Poissonova	SEM	8 Pixel Neighbourhood
Krajina 1	461 sec	583 sec	421 sec	49 sec
Krajina 2	329 sec	426 sec	272 sec	32 sec

1.1.5 Shrnutí

Exemplární metoda funguje dobře pro velké objekty. Pro škrábance je pak vhodná Poissonova metoda. Exemplární metoda využívá syntézu textury. Ke zvýšení efektivity se používá Poissonova metoda. Ke snížení nároků na výpočty se využívá Algoritmus postupné eliminace.

Výše popsaný přístup není jedinou cestou pro řešení inpaintingu. Jak již bylo předznamenáno, existuje mnoho sofistikovaných přístupů. Závěrem tedy uvedeme metody pokročilejšího charakteru, založené na různých modelech.

- Euler's Elastica and Curvature Based Inpaintings [4]
- Image Replacement through Texture Synthesis [5]
- Morphologically Invariant PDE Inpaintings [6]
- Non-Texture Inpainting by Curvature-Driven Diffusions CDD [7]

2 APROXIMACE A EXTRAPOLACE

Obraz jako takový je diskrétní funkce dvou proměnných, souřadnic (x,y). Každé souřadnici tedy odpovídá jedna hodnota pixelu. Pro jednoduchost si můžeme vzít pouze řádek obrazu a mít tak funkci pouze jedné proměnné, souřadnice x. Pokud potřebujeme obraz rozšířit, tedy doplnit hodnoty pixelu za hranici obrazu, musíme zjistit hodnoty této funkce v bodech, kde není definována. K tomuto účelu slouží aproximace a extrapolace, o kterých je pojednáno níže. K dané problematice ještě poznamenáme, že využití všech hodnot v řádku je nevhodné, neboť hodnoty na začátku řádku zřejmě nebudou výraznou měrou souviset s hodnotami na jeho konci. Využijeme proto pouze několik hodnot pixelů na koncovém úseku řádku.

2.1 Aproximační metody

Následující text vychází ze skript FSI VUT v Brně [13]. U některých funkcí se funkční hodnota vypočítá snadno. U některých je zapotřebí použít kalkulačku. Jiné funkce jsou zadány tak složitě, že je lepší najít si jejich hodnoty v tabulce, než je počítat (např. ve statistice). Může se nám stát, že máme funkci, která není zadána žádným předpisem, ale známe pouze její hodnoty v určitých bodech (např. získané měřením nebo konkrétně náš případ, tedy obraz). Pak se naskýtá otázka, jak zjistit hodnotu takové funkce v netabulkovém bodě nebo jak v tomto bodě vypočítat hodnotu derivace. Řešením je nahradit zkoumanou funkci funkcí jinou, která se jí jakýmsi způsobem podobá a se kterou se lépe pracuje. Cílem této kapitoly je ukázat několik možností takovéto náhrady.

Nejčastěji se jako náhradní funkce využívá algebraický polynom, protože jsou s ním jednoduché výpočty, jako např. derivace, integrace atp.

Náhradní funkci vybíráme pomocí různých požadavků. Pokud chceme, aby aproximující funkce měla s funkcí původní v určitých bodech stejné hodnoty, použijeme např. interpolaci. Pokud má aproximující funkce procházet zadaným bodům v jistém smyslu nejblíže, ale přímo jimi procházet nemusí, můžeme použít metodu nejmenších čtverců.

2.1.1 Interpolace algebraickými polynomy

Při interpolaci zní základní úloha takto: Máme $n + 1$ navzájem různých bodů $x_0, x_1, \ldots, x_n$, kterým říkáme uzlové body nebo uzly interpolace a dále funkční hodnoty v těchto bodech $f_0 = f(x_0), f_1 = f(x_1), \ldots, f_n = f(x_n)$. Hledáme polynom $P_n(x)$ stupně nejvýše n takový, že v uzlových bodech nabývá týchž hodnot jako funkce f, tj. $P(x_i) = f_i, i = 0, \ldots, n$.

Poznámka. Někdy se hledá polynom, který má se zadanou funkcí nejen stejné funkční hodnoty v uzlových bodech, ale i stejné hodnoty derivací až do určitého řádu.

Lagrangeův interpolační polynom

Interpolační polynom daný body $[x_i, f_i], i = 0, \dots n$ sestavíme pomocí polynomů $l_i(x)$ takových, že

$$l_i(x_j) = \begin{cases} 1 & \text{pro } i = j \\ 0 & \text{pro } i \neq j \end{cases}$$

Čtenář snadno ověří, že polynom

$$l_0(x) = \frac{(x - x_1)(x - x_2) \dots (x - x_n)}{(x_0 - x_1)(x_0 - x_2) \dots (x_0 - x_n)}$$

má v x_0 hodnotu 1 a v ostatních uzlových bodech hodnotu 0.
Podobně dostaneme i ostatní polynomy $l_i, i = 0, \dots n$:

$$l_i(x) = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_0 - x_1)(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

Interpolační polynom $L_n(x)$ nyní dostaneme snadno jako kombinaci $l_i(x)$:

$$\begin{aligned} L_n(x) = f_0 l_0(x) + f_1 l_1(x) + \dots + f_n l_n(x) &= \\ = f_0 \frac{(x - x_1)(x - x_2) \dots (x - x_n)}{(x_0 - x_1)(x_0 - x_2) \dots (x_0 - x_n)} &+ f_1 \frac{(x - x_0)(x - x_2) \dots (x - x_n)}{(x_1 - x_0)(x_1 - x_2) \dots (x_1 - x_n)} + \dots \\ \dots + f_n \frac{(x - x_0)(x - x_1) \dots (x - x_{n-1})}{(x_n - x_0)(x_n - x_1) \dots (x_n - x_{n-1})} & \end{aligned} \tag{2.1}$$

Interpolační polynom ve tvaru 2.1 se nazývá **Lagrangeův interpolační polynom**.

Newtonův interpolační polynom

Interpolační polynom v Lagrangeově tvaru má tu nevýhodu, že chceme-li přidat další uzlový bod, musíme celý polynom přepočítat znovu. Také výpočet hodnoty tohoto polynomu v určitém bodě je dosti pracný. Proto je někdy výhodnější hledat interpolační polynom v jiném tvaru než 2.1. Jako vhodný se ukazuje tvar

$$N_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) . \tag{2.2}$$

Koeficienty $a_0, a_1, \dots, a_n$ lze získat řešením soustavy rovnic vzniklé rozepsáním podmínek $N_n(x_i) = f(x_i), i = 0, 1, \dots n$, ale přehlednější a méně pracné je vypočítat tyto koeficienty pomocí takzvaných **poměrných diferencí**.

Pro danou funkci f a uzlové body $x_i, i = 0, \ldots, n$ nazveme podíly

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i},\ i = 0, 1, \ldots n-1 \tag{2.3}$$

poměrnými diferencemi prvního řádu.

Pomocí poměrných diferencí prvního řádu definujeme poměrné diference druhého řádu jako

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i},\ i = 0, 1, \ldots, n-2 \tag{2.4}$$

a obecně **poměrné diference k-tého řádu** pro $k \leq n$ definujeme takto:

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i},\ i = 0, \ldots n-k\,. \tag{2.5}$$

Dá se dokázat, že pro koeficienty $a_i, i = 0, 1, \ldots, n$ v 2.2 platí

$$\begin{aligned} a_0 &= f(x_0) \\ a_1 &= f[x_0, x_1] \\ a_2 &= f[x_0, x_1, x_2] \\ &\vdots \\ a_n &= f[x_0, x_1, \ldots, x_n] \end{aligned}$$

Dosazením těchto hodnot do 2.2 dostaneme **Newtonův interpolační polynom**

$$\begin{aligned} N_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots \\ \ldots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1})\,. \end{aligned} \tag{2.6}$$

Poznámka. Newtonův interpolační polynom není vhodné upravovat roznásobováním

Odhad chyby

Nechť interval I obsahuje body $x_0, x_1, \ldots, x_n$ a nechť f je (n+1)-krát diferencovatelná funkce na I. Nechť $P_n(x)$ je interpolační polynom n-tého stupně určený hodnotami funkce f v bodech $x_0, \ldots x_n$. Potom pro libovolné $x \in I$ existuje $\xi \in I$ takové, že pro chybu interpolace $E(x)$ platí

$$E(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\ldots(x - x_n)\,. \tag{2.7}$$

Použití vzorce 2.7 je poněkud problematické, protože bod ξ je pro každé $x \in I$ jiný a jeho nalezení je prakticky nemožné. Chybu interpolace však můžeme alespoň shora odhadnout:
Označíme-li $M_{n+1} = \max_{t \in I} \left| f^{(n+1)}(t) \right|$, platí

$$|E(x)| = |f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |(x - x_0)(x - x_1) \dots (x - x_n)| \ . \tag{2.8}$$

Najít veličinu M_{n+1} však také nemusí být zrovna jednoduché.

Je důležité poznamenat, že v bodech vzdálených uzlovým bodům nabývá výraz $(x - x_0)(x - x_1) \dots (x - x_n)$, který se vyskytuje v odhadu chyby, velkých hodnot. Proto se interpolační polynom pro výpočet přibližných hodnot funkce v takovýchto bodech nehodí.

Aproximace ale v některých případech nemusí být dobrá ani v bodech relativně blízkých uzlovým bodům. Může se totiž stát, že aproximační polynom sice prochází přesně uzlovými body, ale už v bodě mírně vzdáleném od uzlového je jeho hodnota velmi vzdálena od hodnoty původní funkce, protože průběh polynomu mezi uzlovými body jaksi „osciluje“. Situace se mnohdy příliš nezlepší, ani když přidáme více uzlových bodů.

Z tohoto důvodu je někdy vhodné nenahrazovat funkci, zvláště chceme-li ji aproximovat na delším intervalu, jedním interpolačním polynomem, ale interval rozdělit na malé části a na každé z nich funkci nahradit polynomem nízkého stupně. Tímto se zabývá následující kapitola.

2.1.2 Interpolace pomocí splajnů

Základní myšlenka interpolace pomocí splajnů je obdobná, jako u Lagrangeovy interpolace. Máme zadány uzlové body $a = x_0 < x_1 < \ldots < x_n = b$ a funkční hodnoty v nich, které označíme $f_0, f_1, \dots, f_n$. Stejně jako před tím hledáme funkci $S(x)$ takovou, že platí $S(x_i) = f_i, i = 0, 1, \dots, n$, ale tentokrát je funkce $S(x)$ po částech polynom (obecně na každém intervalu $\langle x_i, x_{i+1} \rangle, i = 0, 1, \dots n-1$, jiný) a splňuje určité požadavky hladkosti (tj. spojitosti derivací).

Konkrétně splajnem řádu k pro uzly $a = x_0 < x_1 < \ldots < x_n = b$ rozumíme funkci, která je v každém intervalu $\langle x_i, x_{i+1} \rangle, i = 0, \dots n-1$, polynom stupně k a která má v celém intervalu $\langle a, b \rangle$ spojité derivace až do řádu $k-1$ včetně.[1]

Nejjednodušším příkladem je splajn řádu 1, **lineární splajn**. Funkce je na každém subintervalu $\langle x_i, x_{i+1} \rangle, i = 0, \dots n-1$, nahrazena úsečkou, jejíž rovnice je

$$S_i(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i), \quad x \in \langle x_i, x_{i+1} \rangle \ .$$

[1] Slovo „splajn“ pochází z anglického „spline“, což znamená pružné konstruktérské pravítko. V české literatuře se někdy píše splajn a někdy spline.

U splajnu 1. řádu požadujeme spojitost derivací do řádu 0 včetně, tj. spojitost samotné funkce $S(x)$. Snadno se přesvědčíme, že hodnoty jednotlivých funkcí $S_i(x)$ v krajních bodech příslušného intervalu $\langle x_i, x_{i+1}\rangle$ jsou rovny $f(x_i)$, resp. $f(x_{i+1})$, čímž je zaručeno, že na sebe tyto funkce v uzlových bodech spojitě navazují. Zlepšení aproximace dosáhneme zjemněním intervalů mezi uzlovými body.

Nejčastěji se používají tzv. **kubické splajny**, kdy $k = 3$.

Definice a konstrukce kubického splajnu

Kubický splajn pro funkci f s uzlovými body $x_0, x_1, \ldots, x_n$ je funkce $S(x)$, která je kubický polynom označený $S_i(x)$ na každém subintervalu $\langle x_i, x_{i+1}\rangle$, $i = 0, 1, \ldots n-1$, vyhovuje podmínkám

$$\begin{aligned} S_i(x_i) &= f(x_i), \qquad i = 0, \ldots, n && (2.9)\\ S_i(x_{i+1}) &= S_{i+1}(x_{i+1}), i = 0, \ldots, n-2 && (2.10)\\ S_i'(x_{i+1}) &= S_{i+1}'(x_{i+1}), i = 0, \ldots, n-2 && (2.11)\\ S_i''(x_{i+1}) &= S_{i+1}''(x_{i+1}), i = 0, \ldots, n-2 && (2.12) \end{aligned}$$

a okrajovým podmínkám 1., 2. nebo 3.

1. $S''(x_0) = S''(x_n) = 0$
2. $S''(x_0) = f_0'', S''(x_n) = f_n''$
3. $S'(x_0) = f_0', S'(x_n) = f_n'$

(f_0'', f_n'', f_0' a f_n' jsou předem zadané konstanty).

Podmínky 2.10 znamenají spojitost funkce S v uzlových bodech, podmínky 2.11 a 2.12 spojitost prvních, resp. druhých derivací.

Kubický splajn, splňující okrajové podmínky 1. se nazývá **přirozený kubický splajn**.

Nyní se budeme zabývat tím, jak k zadaným uzlovým bodům a hodnotám funkce v nich sestrojit přirozený kubický splajn. (Splajn vyhovující jiným okrajovým podmínkám by se našel podobně.)

Na jednotlivých intervalech $\langle x_i, x_{i+1}\rangle$, $i = 0, 1, \ldots, n-1$, budeme splajn hledat ve tvaru

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 .$$

Z podmínek 2.9 dostaneme $a_i = f(x_i), i = 0, 1, \ldots, n-1$. Odtud z podmínek 2.10, 2.11, 2.12 a z okrajových podmínek $S_0''(x_0) = S_{n-1}''(x_n) = 0$ lze odvodit soustavu rovnic s neznámými $c_i, i = 0, \ldots, n$

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\left(\frac{\Delta f_i}{h_i} - \frac{\Delta f_{i-1}}{h_{i-1}}\right),\ i = 1, \ldots, n-1 \quad (2.13)$$

$$c_0 = c_n = 0 ,$$

kde $h_i = x_{i+1} - x_i$ a $\Delta f_i = f(x_{i+1}) - f(x_i), i = 0, \ldots, n-1$.

Po rozepsání a dosazení za c_0 a c_n soustava vypadá takto:

$$\begin{aligned}
2(h_0 + h_1)c_1 \quad &+ \quad h_1 c_2 && = 3\left(\frac{\Delta f_1}{h_1} - \frac{\Delta f_0}{h_0}\right) \\
h_1 c_1 \quad &+ \quad 2\left(h_1 + h_2\right)c_2 + h_2 c_3 && = 3\left(\frac{\Delta f_2}{h_2} - \frac{\Delta f_1}{h_1}\right) \\
&\qquad\qquad \ddots && \vdots \\
&\quad h_{n-2}c_{n-2} \qquad + 2\left(h_{n-2} + h_{n-1}\right)c_{n-1} && = 3\left(\frac{\Delta f_{n-1}}{h_{n-1}} - \frac{\Delta f_{n-2}}{h_{n-2}}\right)
\end{aligned} \tag{2.14}$$

Jedná se o třídiagonální soustavu rovnic a lze ji vyřešit např. pomocí Gaussovy eliminační metody přizpůsobené pro třídiagonální soustavu.

Koeficienty b_i a d_i pak dopočítáme pomocí c_i ze vztahů (také odvozených z podmínek $2.9 - 2.12$)

$$b_i = \frac{f(x_{i+1}) - f(x_i)}{h_i} - \frac{c_{i+1} + 2c_i}{3}h_i \quad i = 0, \ldots, n-1 \tag{2.15}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i} \quad i = 0, \ldots, n-1 \tag{2.16}$$

2.1.3 Metoda nejmenších čtverců

V předchozích částech této kapitoly jsme požadovali, aby interpolační polynom, resp. splajn, nabýval v uzlových bodech stejných hodnot jako funkce, kterou se snažíme aproximovat. V případě, že jsou funkční hodnoty získány experimentálně, např. jako výsledky nějakého měření, je interpolace nevhodná. Výsledky jsou totiž zatíženy chybami a interpolační funkce by tyto chyby kopírovala, což je přesně to, co nechceme. Vzhledem k tomu není dobré požadovat, aby aproximační funkce nabývala v uzlových bodech předem daných hodnot. V mnoha případech máme určitou představu o povaze funkce, jejíž hodnoty jsme naměřili, např. se může jednat o lineární nebo kvadratickou závislost. Pak hledáme mezi všemi funkcemi tohoto typu takovou, která prochází k zadaným bodům v jistém smyslu nejblíže. Obrazová funkce je stejně jako funkce získaná měřením zadána hodnotami v uzlových bodech. Informaci o povaze funkce ale bohužel nemáme, neboť průběh hodnot pixelů na řádku nebývá většinou jednoduchá funkce.

Formulace problému

Jsou dány body $x_i, i = 0, \ldots n$ a funkční hodnoty v nich y_i. Dále jsou dány funkce $\varphi_i, i = 0, \ldots, m$, $m < n$. Mezi všemi funkcemi tvaru

$$P_m(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + \ldots + c_m\varphi_m(x)\,, \tag{2.17}$$

$c_0, \ldots, c_m$ jsou reálná čísla, hledáme takovou, pro niž veličina

$$\rho^2(c_0, \ldots c_m) = \sum_{i=0}^{n}(y_i - P_m(x_i))^2\,, \tag{2.18}$$

kterou nazýváme **kvadratická odchylka**, nabývá minimální hodnoty. (Kvadratická odchylka ρ^2 udává součet obsahů čtverců se stranami o délkách $|y_i - P_m(x_i)|$, $i = 0, \ldots, n$. Odtud tedy pochází název metody.)

Takovou funkci pak nazýváme nejlepší aproximací experimentálních dat $y_0, \ldots y_n$ v dané třídě funkcí ve smyslu metody nejmenších čtverců.

Nalezení nejlepší aproximace

Protože body $[x_i, y_i], i = 0, \ldots, n$ a funkce $\varphi_i, i = 0, \ldots, m$, jsou dány, kvadratická odchylka

$$\rho^2 = \sum_{i=0}^{n}(y_i - c_0\varphi_0(x_i) - c_1\varphi_1(x_i) - \ldots - c_m\varphi_m(x_i))^2 \tag{2.19}$$

závisí pouze na koeficientech $c_0, \ldots, c_m$. Z diferenciálního počtu funkcí více proměnných je známo, že nutnou podmínkou pro to, aby $\rho^2(c_0 \ldots, c_m)$ nabývala minima, je splnění rovnic

$$\frac{\partial}{\partial c_j}(\rho^2) = \frac{\partial}{\partial c_j}\left[\sum_{i=0}^{n}(y_i - c_0\varphi_0(x_i) - c_1\varphi_1(x_i) - \ldots - c_m\varphi_m(x_i))^2\right] = 0, j = 0, \ldots, m\,. \tag{2.20}$$

Zderivováním dostaneme

$$\sum_{i=0}^{n} 2(y_i - c_0\varphi_0(x_i) - c_1\varphi_1(x_i) - \ldots - c_m\varphi_m(x_i))(-\varphi_j(x_i)) = 0, j = 0, \ldots, m\,. \tag{2.21}$$

Rovnice vydělíme -2 a rozdělíme na jednotlivé sumy:

$$\sum_{i=0}^{n} y_i\varphi_j(x_i) - \sum_{i=0}^{n} c_0\varphi_0(x_i)\varphi_j(x_i) - \ldots - \sum_{i=0}^{n} c_m\varphi_m(x_i)\varphi_j(x_i) \quad j = 0, \ldots, m\,. \tag{2.22}$$

Z každé sumy můžeme vytknout odpovídající koeficient c_k. Snadnou úpravou pak dostaneme tzv. normální rovnice pro neznámé $c_0, \ldots, c_m$:

$$c_0\sum_{i=0}^{n}\varphi_0(x_i)\varphi_j(x_i) + \ldots + c_m\sum_{i=0}^{n}\varphi_m(x_i)\varphi_j(x_i) = \sum_{i=0}^{n} y_i\varphi_j(x_i) \quad j = 0, \ldots, m\,. \tag{2.23}$$

Tato soustava rovnic po rozepsání vypadá takto:

$$\begin{aligned}
c_0\sum_{i=0}^{n}\varphi_0^2(x_i) &+ c_1\sum_{i=0}^{n}\varphi_1(x_i)\varphi_0(x_i) + \ldots + c_m\sum_{i=0}^{n}\varphi_m(x_i)\varphi_0(x_i) &= \sum_{i=0}^{n} y_i\varphi_0(x_i)\\
c_0\sum_{i=0}^{n}\varphi_0(x_i)\varphi_1(x_i) &+ c_1\sum_{i=0}^{n}\varphi_1^2(x_i) + \ldots + c_m\sum_{i=0}^{n}\varphi_m(x_i)\varphi_1(x_i) &= \sum_{i=0}^{n} y_i\varphi_1(x_i)\\
&\qquad\qquad \ddots \qquad\qquad \vdots &\\
c_0\sum_{i=0}^{n}\varphi_0(x_i)\varphi_m(x_i) &+ c_1\sum_{i=0}^{n}\varphi_1(x_i)\varphi_m(x_i) + \ldots + c_m\sum_{i=0}^{n}\varphi_m^2(x_i) &= \sum_{i=0}^{n} y_i\varphi_m(x_i)
\end{aligned} \tag{2.24}$$

Získaná soustava rovnic vypadá poněkud hrozivě a nepřehledně, ale s konkrétními funkcemi φ_i se situace vyjasní.

Aproximace metodou nejmenších čtverců algebraickými polynomy

Velmi častá volba funkcí φ_i je $\varphi_i(x) = x^i, i = 0, 1, \ldots, m$, tj.

$$\varphi_0(x) = 1, \varphi_1(x) = x, \ldots, \varphi_m(x) = x^m\,.$$

Aproximující funkce P_m je pak tvaru

$$P_m(x) = c_0 + c_1x + \ldots + c_mx^m$$

a jednotlivé sumy v soustavě normálních rovnic vyjdou

$$\sum_{i=0}^{n}\varphi_0^2(x_i) = \sum_{i=0}^{n} 1 = \underbrace{1+1+\ldots+1}_{n+1} = n+1,\ \sum_{i=0}^{n}\varphi_1(x_i)\varphi_0(x_i) = \sum_{i=0}^{n} x_i\cdot 1 = \sum_{i=0}^{n} x_i, \ldots$$

obecně

$$\sum_{i=0}^{n}\varphi_k(x_i)\varphi_l(x_i) = \sum_{i=0}^{n} x_i^k\cdot x_i^l = \sum_{i=0}^{n} x_i^{k+l}, \quad k,l = 0,\ldots,m\,.$$

Soustava normálních rovnic pak vypadá následovně

$$\begin{aligned}
c_0(n+1) &+ c_1\sum_{i=0}^{n} x_i + \ldots + c_m\sum_{i=0}^{n} x_i^m &= \sum_{i=0}^{n} y_i\\
c_0\sum_{i=0}^{n} x_i &+ c_1\sum_{i=0}^{n} x_i^2 + \ldots + c_m\sum_{i=0}^{n} x_i^{m+1} &= \sum_{i=0}^{n} x_iy_i\\
&\qquad\qquad \vdots &\\
c_0\sum_{i=0}^{n} x_i^m &+ c_1\sum_{i=0}^{n} x_i^{m+1} + \ldots + c_m\sum_{i=0}^{n} x_i^{2m} &= \sum_{i=0}^{n} x_i^my_i
\end{aligned} \tag{2.25}$$

Speciálně pro aproximaci přímkou $P_1(x) = c_0 + c_1x$ dostaneme soustavu

$$\begin{aligned}
c_0(n+1) &+ c_1\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_i\\
c_0\sum_{i=0}^{n} x_i &+ c_1\sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} x_iy_i
\end{aligned} \tag{2.26}$$

a pro aproximaci parabolou $P_2(x) = c_0 + c_1 x + c_2 x^2$ soustavu

$$\begin{aligned} c_0(n+1) &+ c_1 \sum_{i=0}^{n} x_i &+ c_2 \sum_{i=0}^{n} x_i^2 &= \sum_{i=0}^{n} y_i \\ c_0 \sum_{i=0}^{n} x_i &+ c_1 \sum_{i=0}^{n} x_i^2 &+ c_2 \sum_{i=0}^{n} x_i^3 &= \sum_{i=0}^{n} x_i y_i \\ c_0 \sum_{i=0}^{n} x_i^2 &+ c_1 \sum_{i=0}^{n} x_i^3 &+ c_2 \sum_{i=0}^{n} x_i^4 &= \sum_{i=0}^{n} x_i^2 y_i \,. \end{aligned} \tag{2.27}$$

2.2 Prodloužení konstantou

Ve speciálních případech můžeme pro doplnění pixelů za hranicí obrazu namísto aproximace použít prodloužení konstantou. Jednoduše nakopírujeme poslední pixel tolikrát, kolikrát je potřeba. Tento způsob řešení je však vhodný pouze pro velmi málo typů obrazu, např. pro jednobarevná pozadí nebo pruhy a zmiňujeme ho spíše pro úplnost.

2.3 Zrcadlení

Dalším přístupem k doplňování pixelů vně obrazu může být zrcadlení objektů. Uvažujme případ, kdy máme na okraji snímku např. automobil z čelního pohledu. Není však v záběru celý, nýbrž jen jeho polovina. Za tohoto předpokladu je možné doplnit zbývající část automobilu pomocí zrcadlení zachycené poloviny na druhou stranu podle osy souměrnosti automobilu. Pro dobrý výsledek je však nezbytné, aby se na snímku nacházela přesně nebo více než polovina automobilu. Jinak bychom nemohli zrcadlení provést. Další úskalí, které je zřejmé, je nutnost nalezení oné osy souměrnosti, kterou je potřeba v obrazu detekovat. A hlavně vůbec zjistit, že se jedná o objekt souměrný a že je tudíž možné zrcadlení aplikovat.

2.4 Shrnutí

Aproximace funkce spočívá v nahrazení zkoumané funkce$f$ jednodušší funkcí, která nabývá přibližně stejných hodnot jako funkce f a se kterou se lépe pracuje.

U interpolace hledáme funkci, která má s f společné funkční hodnoty v uzlových bodech $x_0, x_1 \ldots x_n$. Nejčastěji to bývá aproximační polynom nebo splajn.

Interpolační polynom $P_n(x)$ je algebraický polynom stupně nanejvýš n, pro nějž platí $P(x_i) = f(x_i), i = 0, 1, \ldots, n$. Interpolační polynom pro zadané body existuje vždy právě jeden, ale můžeme jej vyjádřit v různém tvaru.

Existují speciální tvary interpolačních polynomů pro ekvidistantní uzly. Ty se vyznačují tím, že krok mezi všemi dvojicemi sousedních uzlů mají konstantní.

Lagrangeův interpolační polynom sestavíme přímo ze zadaných uzlů a funkčních hodnot v nich. Pro konstrukci Newtonova interpolačního polynomu musíme nejprve vypočítat poměrné (pokud jde o neekvidistantní uzly) nebo obyčejné (jde-li o ekvidistantní uzly) diference a interpolační polynom pak sestavíme pomocí nich. Newtonův interpolační polynom má narozdíl od Lagrangeova tu výhodu, že se do něj snadněji dosazuje a snadněji se dá přidat další uzel.

Za příznivých okolností platí $f(x) \doteq P_n(x)$. Pokud ale použijeme příliš mnoho uzlových bodů, interpolační polynom může (ale nemusí) začít „oscilovat“. Z tohoto důvodu je pro aproximaci funkce na dlouhém intervalu lepší splajn.

Splajn $S(x)$ je rovněž funkce, pro niž platí v neuzlových bodech $S(x_i) = f(x_i)$, $i = 0, 1, \ldots, n$, ale na rozdíl od interpolačního polynomu je to funkce definovaná po částech. Je dána jiným předpisem na každém z intervalů $\langle x_i, x_{i+1}\rangle$, $i = 0, 1, \ldots, n-1$. Nejčastěji se používá tzv. přirozený kubický splajn. To je funkce na každém intervalu $\langle x_i, x_{i+1}\rangle$ polynom třetího stupně $S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$. Jednotlivé polynomy S_i a S_{i+1} na sebe musí v bodě x_{i+1} (to je bod, kde se stýkají jejich definiční obory) spojitě navazovat až do druhé derivace včetně. Navíc musí platit okrajové podmínky $S_0'' = S_{n-1}''(x_n) = 0$.

Při výpočtu splajnu nejprve najdeme koeficienty c_i jako řešení jisté soustavy lineárních rovnic. Koeficienty b_i a d_i pak vypočteme pomocí nich. Pro koeficienty a_i platí $a_i = f_i$.

Metoda nejmenších čtverců se používá především v případě, kdy máme hodnoty $[x_i, y_i], i = 0, 1, \ldots, n$, získané nějakým měřením (tj. zatížené chybami) a máme určitou představu o povaze funkční závislosti y na x. Předpokládáme, že tato funkční závislost je typu $y = c_1\varphi_1(x) + \ldots + c_m\varphi_m(x)$, kde $\varphi_i, i = 0, \ldots, m$, jsou známé funkce. Mezi všemi funkcemi tohoto známého typu hledáme tu, pro kterou je minimální tzv. kvadratická odchylka. Nalezení této funkce spočívá v nalezení hodnot koeficientů $c_i, i = 0, \ldots, m$. Ty najdeme jako řešení tzv. soustavy normálních rovnic. Pro aproximaci algebraickým polynomem je tvar soustavy známý. Speciálně pro polynom prvního stupně, to je přímku, je to 2.26 a pro polynom druhého stupně, to je parabolu, 2.27. Chceme-li použít jiný typ funkcí, musíme dosadit do obecného tvaru soustavy 2.24.

3 WAVELETOVÁ TRANSFORMACE

Kapitola o Waveletové transformaci je převzata z [14], [15] a [16]. Bude zde postupně popsán multirozklad, definována spojitá waveletová transformace a její vlastnosti. Podíváme se na teorii konstrukce ortonormálních waveletů, vysvětlíme Diskrétní waveletovou transformaci (DWT) a Diskrétní waveletovou transformaci s diskrétním časem (DTWT). A nakonec bude na obrázcích vysvětlen algoritmus výpočtu diskrétní waveletové transformace s diskrétním časem a dvourozměrná waveletová transformace.

Souvislost waveletové transformace s asi nejznámější Fourierovou transformací je následující. Zobecněné Fourierovy transformace mají různé vlastnosti dané použitou bází a z toho plynoucí aplikace. Zatímco u řady těchto jednorozměrných zobecněných FT dochází k transformaci prostoru s určitým fyzikální rozměrem (např. $[r]$ u klasické FT) do prostoru s fyzikálním rozměrem jiným (např. $[r^{-1}]$ u klasické FT), tak u waveletové neboli vlnkové transformace (WT), využívající bázi odvozenou od základní funkce pomocí posunutí a změny měřítka, dochází k transformaci jednorozměrného prostoru do dvourozměrného, majícího však stejný rozměr fyzikální. Waveletová analýza je tudíž speciálním případem Fourierovy analýzy.

K popisu WT potřebujeme složitější matematický aparát, jehož základem bude tzv. víceúrovňová analýza (multirozklad $\mathbf{L}^2(\mathbb{R})$ neboli multiresolution analysis - MRA).

Pokusíme se vysvětlit základy tohoto procesu na jednoduchém příkladu. Představme si, že v supermarketu máme stojan s policemi (nazveme jej Hilbertovým prostorem $\mathbf{L}^2(\Omega)$), označíme podlahu jako nulovou hladinu $m = 0$. Máme k dispozici zásilku zboží (funkce $f(t)$), např. zásilka míčů různé velikosti, stojan je sestaven z jednotlivých polic, každá police tvoří jednu hladinu (V_m). Na každé polici jsou kulaté otvory (jednotlivé bázové elementy ϕ_{mn}, m je index police, n je počet otvorů na polici), průměry otvorů jsou stejné na jedné polici (na jedné hladině), ale při přechodu z jedné police na druhou se mění dle vztahu $2^m, m = 0, 1, 2, \ldots$, tj. otvory na nižší polici jsou dvakrát menší než o jednu výše. Úkolem bude najít optimální rozklad zboží na policích, tj. aby každý míč byl uložen na příslušné místo, odpovídající jeho průměru.

Řešení: Je-li míč menší než otvor, pak tento míč spadne přes otvor na příslušnou polici (na příslušnou hladinu), kde otvor bude menší než průměr míče. Uložit tento míč na nižší polici je ekonomicky nevýhodné - tam může být umístěn míč o menším průměru. Proceduru právě tohoto rozložení nazýváme víceúrovňovou analýzou. Díváme-li se do dalekohledu, mikroskopu, fotoaparátu, snažíme se zachytit ostré a zároveň dostatečně velké zobrazení našeho objektu, přitom automaticky provádíme podobnou analýzu.

3.1 Multirozklad - MRA

Multirozkladem $\mathbf{L}^2(\mathbb{R}^n)$ (víceúrovňovou analýzou) budeme nazývat neklesající posloupnost uzavřených škálovacích podprostorů $V_m \in \mathbf{L}^2(\mathbb{R}^n), m \in \mathbb{Z}$ pro něž platí následující podmínky:

1. $\ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots$ tj. $V_m \subset V_{m+1}, \forall m \in \mathbb{Z}$
2. $\bigcap_{m\in\mathbb{Z}} V_m = \{0\}$
3. $\bigcup_{m\in\mathbb{Z}} V_m$ je hustý a představuje $\mathbf{L}^2(\mathbb{R}^n)$, tj. $\overline{\bigcup_{m\in\mathbb{Z}} V_m} = \mathbf{L}^2(\mathbb{R}^n)$
4. $f(\mathbf{t}) \in V_m \Leftrightarrow f(2\mathbf{t}) \in V_{m+1},\ \mathbf{t} \in \mathbb{R}^n$
5. existuje funkce $\phi \in V_0$ taková, že $\{\phi(\mathrm{t} - \mathrm{n})\}_{n\in\mathbb{Z}}$ je ortonormální bází V_0. Funkce $\phi \in V_0$ se nazývá škálovací funkce nebo základní škálová funkce resp. otcovský wavelet.

Formálně bychom mohli MRA interpretovat následujícím způsobem. Bod 1 představuje rozdělení stojanu na jednotlivé police, někteří autoři zapisují $V_{m+1} \subset V_m$, $\forall m \in \mathbb{Z}$, my této indexace využijeme v budoucnu u DWT. Bod 3 říká, že sjednocením získáme celý stojan, tj. $V_\infty = \mathbf{L}^2(\mathbb{R}^n)$ a $V_{-\infty} = \{0\}$, tzn. neexistuje žádné zboží, které by se udržovalo na horní polici. Bod 4 znamená, že funkce $f \in V_{m+1}$ obsahuje dvakrát více bodů než $f \in V_m$, což odpovídá proceduře přeložení míče na nejbližší nižší polici za podmínky, že průměr otvoru je dvakrát menší, než na předchozí polici. $\phi \in V_0$ v bodě 5 slouží pro analýzu jednotlivých škálovaných podprostorů V_m - police v horizontálním směru vyplní celý prostor bez překrývání a mezer, tedy bázi prostoru $V_1 : \{\phi(2t - n)\}_{n\in\mathbb{Z}}$ dostaneme z elementů báze prostoru $V_0 : \{\phi(t - n)\}_{n\in\mathbb{Z}}$ jednoduchým dvojnásobným zmenšením posledních.

Existují další definice, které se vztahují k biortogonalizaci waveletů, k waveletům v prostoru $\mathbf{L}^p$, k waveletům na distribucích apod. Základem pro vytvoření diskrétní waveletové transformace (DWT) je dilatační rovnice tzv. rovnice soběpodobnosti - škálovací rovnice. Formální řešení této rovnice můžeme sestavit ve tvaru Fourierova integrálu, avšak analýza vznikajících funkcí není vůbec jednoduchá.

3.2 Definice spojité Waveletové transformace

Nechť $f(t), \psi(t) \in \mathbf{L}^2\mathbb{R}$. Waveletovou transformaci funkce $f(t)$ pak definujeme

$$WT(f) = F(a,b) = \hat{f}(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} f(t)\overline{\psi\left(\frac{t-b}{a}\right)}dt, \tag{3.1}$$

kde $a \in \mathbb{R}\setminus\{0\}$ je tzv. dilatační škálový parametr, $b \in \mathbb{R}$ je translační parametr, $\psi(t)$ je mateřský wavelet nebo jen wavelet splňující

$$\int_{-\infty}^{\infty} \psi(t)dt = 0\,. \tag{3.2}$$

Zpětná (inverzní) waveletová transformace je pak dána vztahem

$$WT^{-1}(F) = f(t) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} F(a,b)\psi \left(\frac{t-b}{a} \right) da \right) db. \tag{3.3}$$

Základní mateřský wavelet si můžeme představit jako „vlnku“ nabývající kladných a záporných hodnot, která nemusí být vůbec symetrická a může být vytvořena i pomocí nelineárních kombinací otcovského waveletu. Obraz F je skalární součin funkce f s dilatacemi a translacemi okna ψ, tj.

$$F(a,b) = \left\langle f(t), \frac{1}{\sqrt{a}} \psi \left(\frac{t-b}{a} \right) \right\rangle . \tag{3.4}$$

3.3 Vlastnosti Waveletové transformace

Koeficienty WT obsahují informaci jak o analyzované funkci, tak i o waveletu použitém při analýze. Nechť $WT(f(t)) = F(a,b)$, pak některé vlastnosti WT jsou nezávislé na typu waveletu:

1. linearita $WT(\alpha f_1 + \beta f_2) = \alpha WT(f_1) + \beta WT(f_2) = \alpha F_1(a,b) + \beta F_2(a,b)$,
2. invariance vzhledem k posunutí $WT(f(t-b_0)) = F(a, b-b_0)$,
3. invariance vzhledem k dilataci $WT\left(f\left(\frac{t}{a_0}\right)\right) = \frac{1}{a_0} F\left(\frac{a}{a_0}, \frac{b}{a_0}\right)$,
4. analogie Parsevalovy věty v případě ortogonální waveletovské báze

$$\int_{-\infty}^{\infty} f_1(t)\overline{f}_2(t)dt = C_\psi^{-1} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_1(a,b)\overline{F}_2(a,b)a^{-2}da\,db \Rightarrow$$

energie signálu (funkce)

$$\int_{-\infty}^{\infty} |f(t)|^2 \, dt = C_\psi^{-1} \int_{-\infty}^{\infty} |F(a,b)|^2 \, a^{-2} da\, db.$$

3.4 Waveletová transformace - konstrukce ortonormálních waveletů

Matematická konstrukce ortonormálních waveletů s kompaktním nosičem byla provedena Ingrid Daubechies (1988) s použitím teorie MRA. Postup tvorby báze pomocí MRA je následující:

- Nechť P_m znamená ortogonální projekci f do V_m a D_{2^m} dilatační operátor, tj. $f(.) \in D_{2^m}V_n \Leftrightarrow f(2^m.) \in V_{m+n}$. S rostoucím m pak $P_m f$ lépe aproximuje f, až nakonec

$$\lim_{m\to\infty} P_m f = f.$$

Prostor V_m je tvořen škálovými funkcemi $\{\phi_{mn}\}_{\forall n}, \forall m \in \mathbb{Z}$.

- Jelikož V_m je obsažen ve V_{m+1}, můžeme definovat W_m jako m-tý waveletový prostor obsahující waveletové funkce $\{\psi_{mn}\}_{\forall n}, \forall m \in \mathbb{Z}$ tak, aby byl ortogonálním doplňkem V_m do V_{m+1}, tj.

  $$V_{m+1} = V_m \oplus W_m,$$

  Q_m je projekční operátor do W_m - škálované verze W_0, kde $f(.) \in W_m \Leftrightarrow f(2^{-m}.) \in W_0$. Tedy obdobně W_m je tvořen waveletovými funkcemi $\{\psi_{mn}\}_{\forall n}$, $\forall m \in \mathbb{Z}$. Pak

  $$P_{m+1} = P_m \oplus Q_m$$

  je projekční operátor do V_{m+1}.
- Základní vlastnost MRA je to, že umožňuje sestavit ortonormální waveletovskou bázi $\{\psi_{mn}(t)\}_{\forall n}, \forall m \in \mathbb{Z}$, kde $\psi_{mn}(t) = 2^{-\frac{m}{2}}\psi(2^{-m}t - n), m, n \in (Z)$ tak, že pro každou $f(t) \in \mathbf{L}^2(\mathbb{R})$ platí:

  $$P_{m+1}f = P_m f + Q_m f = \sum_{n\in\mathbb{Z}} a_{mn}\phi_{mn} + \sum_{n\in\mathbb{Z}} d_{mn}\psi_{mn} =$$

  $$= \sum_{n\in\mathbb{Z}} \langle f, \phi_{mn}\rangle \phi_{mn} + \sum_{n\in\mathbb{Z}} \langle f, \psi_{mn}\rangle \psi_{mn}.$$
- Koeficienty a_{mn} nazveme aproximačními, trendovými, škálovými, nebo nízkofrekvenčními, koeficienty d_{mn} pak detailními, doplňkovými, waveletovými nebo vysokofrekvenčními.
- Hledáme funkci $\psi \in W_0$ tak, aby $\{\psi(t-n)\}_n$ tvořila ortonormální bázi W_0, $W_{m+1} = D_2 W_m$ a $\{D_{2^m}\psi(t-n)\}_n$ byla ortonormální bází W_m. Jelikož $W_m \perp V_m$, $V_{m+1} = V_m \oplus W_m$ a $\mathbf{L}^2 = \overline{\bigcup_{m\in\mathbb{Z}} V_m}, W_{m+1} \perp W_m$ a $\mathbf{L^2} = \oplus_{m\in\mathbb{Z}} W_m$, pak $\{D_{2^m}\psi(t-n)\}_{nm}$ je ortonormální bází $\mathbf{L^2}$. Ortonormalita je zde zaručena na jednotlivých úrovních m:

  $$\langle \phi_{mk}, \phi_{ml}\rangle = \delta_{kl} = \begin{cases} 1, & k = l \\ 0, & k \neq l \end{cases}.$$

  Pro skalární součin mezi sousedními úrovněmi platí

  $$\langle \phi_{mk}, \phi_{m+1,l}\rangle = h_{l-2k}, \quad k, l \in \mathbb{Z}, \quad \sum_{\forall k} h_k^2 = 1.$$
- Konstrukce ψ je dána následující procedurou. Nechť $\mathbf{l^2}$ je diskrétní analog prostoru $\mathbf{L^2}(\mathbb{R})$. Je-li $\phi \in V_0 \subset V_1$ a $\{\phi(2t-n)\}$ je ortonormální bází V_1, pak posloupnost koeficientů $h_n \in \mathbf{l^2}$ splňuje **dilatační rovnici**

  $$\phi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} h_n \phi(2t-n).$$

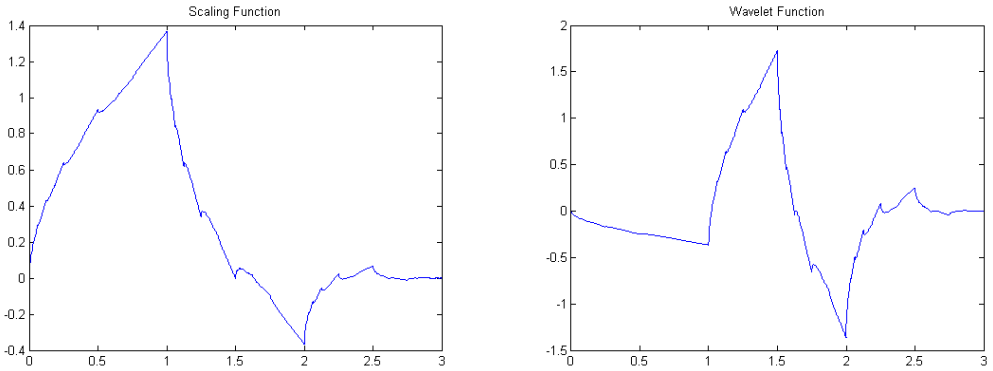
Vyřešení této rovnice odstartovalo konstrukci ortonormálních waveletů. Zde h_n jsou škálovací filtrační koeficienty zaručující ortonormalitu, má-li $\phi(t)$ kompaktní nosič, pak počet těchto nenulových koeficientů je nenulový.

- Definujme
$$\psi(t) = \sqrt{2} \sum_{n \in \mathbb{Z}} g_n \phi(2t - n),$$
pak právě wavelety $\psi_{mn} = 2^{-\frac{m}{2}} \psi(2^{-m}t - n), m, n \in \mathbb{Z}$ tvoří ortonormální waveletové báze prostorů W_m, které nazýváme Daubechies (tyto wavelety nemají žádné osy symetrie, ortonormalita waveletů je zaručena i mezi různými úrovněmi m). Pro ortogonální báze
$$\phi(t) = \sum_{n \in \mathbb{Z}} h_n \phi(2t - n), \psi(t) = \sum_{n \in \mathbb{Z}} g_n \phi(2t - n).$$

Některé další podmínky pro sestavení waveletů (nejsou nutné):

- $\int_{-\infty}^{+\infty} \phi(t)dt = 1$, zde ϕ lze chápat jako prostorovou hustotu rozdělení pravděpodobnosti náhodné veličiny t za podmínky, že $\phi(t) \geq 0, t \in \mathbb{R}$. Uvažujeme-li, že střední hodnota náhodné veličiny t je 0 a její rozptyl 1, pak $\int_{-\infty}^{+\infty} t\phi(t)dt = 0$, $\int_{-\infty}^{+\infty} t^2\phi(t)dt = 1$,
- $\int_{-\infty}^{+\infty} t^k\psi(t)dt = 0,\ k = 1, 2, \ldots$ nulovost momentů k-tého stupně,
- pro $\psi(t) \geq 0 : \int_{-\infty}^{+\infty} \psi(t)dt = 1$,
- podmínka na symetrii apod.

Budeme se snažit nalézt škálovací funkci tvořící ortonormální multirozklad.



Obr. 3.1: Ukázka škálové a waveletové funkce pro Daubechies Wavelet Db2.

3.5 Diskrétní waveletová transformace

Podle [16] lze rozlišit tři druhy waveletových transformací. Spojitou waveletovou transformaci, u níž jsou spojité jak změna měřítka a posunutí, tak i vstupní signál a

waveletové funkce. Popsána byla již výše. Dále pak diskrétní waveletovou transformaci (DWT), kdy je spojitý vstupní signál a waveletové funkce, ale změna měřítka a posuny jsou prováděny diskrétně, a nakonec diskrétní waveletová transformace s diskrétním časem (DTWT), u které jsou vstupní signál i waveletové funkce diskrétní signály a změna měřítka a posuny jsou prováděny rovněž diskrétně. Následující text pochází z [15] a věnuje se zbývajícím dvěma typům transformací.

Při použití waveletové transformace k vyjádření obrazových dat pracujeme s konečným diskrétním signálem, a proto použijeme diskrétní waveletovou transformaci s diskrétním časem. Parametry a a b se mění diskrétně tak, že

$$a = 2^j \tag{3.5}$$

$$b = k \cdot 2^j = k \cdot a \quad \text{pro } j, k \in \mathbb{Z} \text{ a } j \geq 1. \tag{3.6}$$

Měřítko a je vzorkováno v dyadické posloupnosti, zatímco časový posuv rovnoměrně. Je možné ukázat, že vzorky spektra v těchto bodech nesou úplnou informaci o původním signálu, který tak může být přesně rekonstruován.

Dyadická diskrétní waveletová transformace s diskrétním časem je definována takto

$$S_{DTWT}(j,k) = \frac{1}{\sqrt{2^a}} \sum_{n=-\infty}^{\infty} f(n)\psi * (2^{-j}n - k). \tag{3.7}$$

DTWT je unitární transformace, a tudíž ji můžeme vyjádřit pomocí ortogonální matice $\mathbf{W}$ $n \times n$. Máme-li vektor délky n, $y = (y_1, \ldots, y_n)^{\mathrm{T}}$, jeho waveletovou transformaci $d = (d_1, \ldots, d_n)^{\mathrm{T}}$ získáme, když

$$\mathbf{d} = \mathbf{W}\mathbf{y}. \tag{3.8}$$

Díky tomu, že matice $\mathbf{W}$ je ortogonální, inverzní diskrétní waveletová transformace může být vyjádřena jako

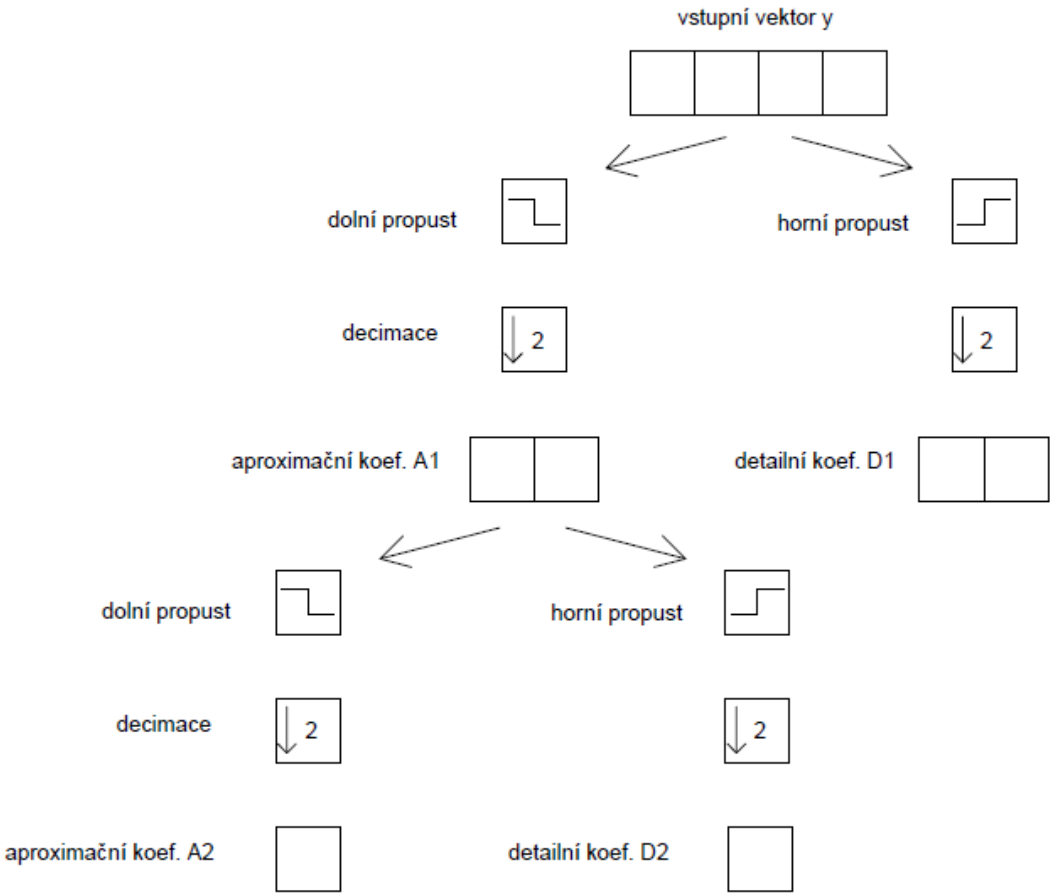
$$\mathbf{W}^{-1}\mathbf{d} = \mathbf{W}^{\mathrm{T}}\mathbf{d} = \mathbf{y}. \tag{3.9}$$

3.6 Algoritmus výpočtu DTWT

Vzhledem k výpočetně náročnému násobení vstupního vektoru y délky n ortogonální maticí $\mathbf{W}$ řádu $n \times n$ je v praxi výhodnější použít pyramidový rekurzivní algoritmus, představený S. Mallatem [17], využívající k analýze a syntéze signálu banku filtrů s vlastností perfektní rekonstrukce, tzn., že aliasing vzniklý při dekompozici se při rekonstrukci vyruší.

Filtrace se provádí pomocí čtyř filtrů. Jsou to syntetizující dolní propust (DP) a syntetizující horní propust (HP) a rekonstrukční DP a HP. Jde vždy o speciální kvadraturní zrcadlové filtry, jejichž moduly kmitočtových charakteristik jsou navzájem zrcadlově obrácené podle $f/f_{vz} = 1/4$.

Vektor vstupu y filtrujeme pomocí filtru typu dolní propust a zároveň filtrem typu horní propust. Obě výsledné posloupnosti decimujeme, tzn., že vypustíme každý druhý vzorek, a tím získáváme dvě nové posloupnosti s přibližně poloviční velikostí původního vstupního vektoru y. Viz obr. 3.2.



Obr. 3.2: Waveletová dekompozice signálu y pomocí algoritmu S. Mallata. Hloubka dekompozice je 2.

Koeficienty získané filtrem typu horní propust nazýváme detailními waveletovými koeficienty. Koeficienty získané filtrem typu dolní propust nazýváme aproximačními koeficienty. Výše popsaným postupem jsme provedli jeden krok waveletové dekompozice. Další krok provedeme obdobně jen s tím rozdílem, že namísto vstupního vektoru y přivádíme na vstup obou filtrů vektor aproximačních koeficientů s tím, že vektor detailních koeficientů zachováme. Počet takto provedených kroků nazýváme

hloubkou dekompozice. Např. pro čtyřprvkový vstupní vektor výše, je maximální hloubka dekompozice 2. Obecně pro vstupní vektor s délkou n platí, že maximální hloubka dekompozice je

$$J_{max} \leq \log_2 n. \tag{3.10}$$

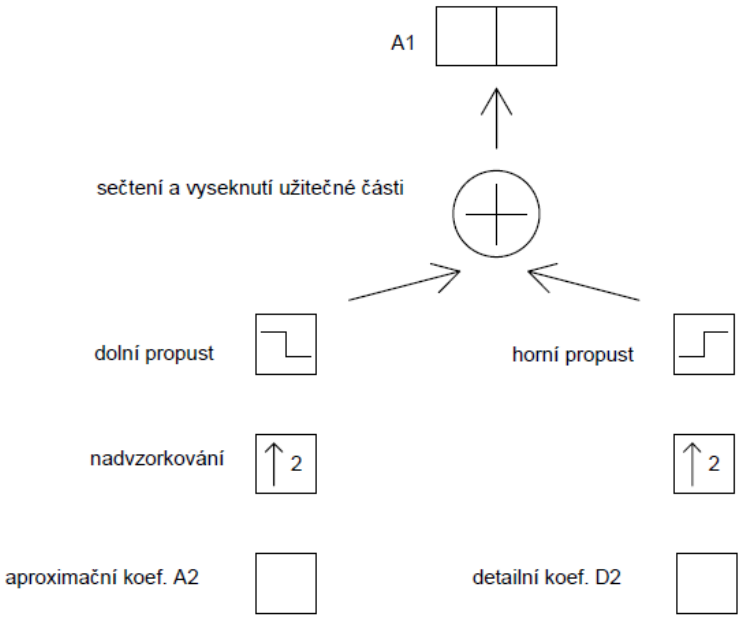
Mallatovým algoritmem můžeme vstupní signál rozložit na detailní a aproximační koeficienty, přitom nedochází k žádné ztrátě informace. Proces dekompozice je možné zastavit ve kterékoli úrovni, a tím je umožněna tzv. neúplná diskrétní waveletová transformace. Tento fakt je další výhodou realizace waveletové transformace pomocí zrcadlově kvadraturních filtrů.

Dekomponovaný signál je popsán svým aproximačním vektorem A_k a všemi vektory detailů D_k až D_1. Není tedy problém z něj vytvořit zpět původní signál, a to přesně opačným postupem. Ukážeme, že na kterékoli úrovni dekompozice k lze získat aproximační koeficienty vyšší úrovně $k-1$ z koeficientů A_k a D_k. Oba vektory jsou nejprve nadvzorkovány. To znamená, že do vektorů aproximačních a detailních koeficientů vkládáme nuly a zvyšujeme tedy jejich délku na dvojnásobek. Následně filtrujeme filtry typu dolní a horní propust, které jsou inverzní k původním dekompozičním filtrům. Výsledky sečteme a vysekneme pouze potřebnou prostřední část. Naznačuje to obr. 3.3. Postup opakujeme tolikrát, kolik je hloubka dekompozice. Nakonec dostaneme originální signál.

3.7 Dvourozměrná waveletová transformace

Tato práce se zabývá zpracováním obrazu. Protože obraz je dvourozměrný signál, je potřeba použít dvourozměrnou diskrétní waveletovou transformaci s diskrétním časem. Dekompozici budeme provádět podobným algoritmem, jaký jsme popsali již výše. Rozdíl je v tom, že obraz budeme filtrovat zvlášť po řádcích a zvlášť po sloupcích. Obecně je jedno, v jakém pořadí. Tím vzniknou čtyři druhy koeficientů, aproximační koeficienty, detailní horizontální koeficienty, detailní vertikální koeficienty a detailní diagonální koeficienty. Situaci pomůže pochopit obr. 3.4.

Data vstupního obrazu projdou filtry typu DP a HP. Následuje decimace, při zpracování po řádcích je vynechán každý druhý prvek v řádku. Výsledná obrazová data projdou znovu filtry typu DP a HP. Nyní se ale zpracovává po sloupcích a je vynechán každý druhý prvek ve sloupci. Po jednom kroku dekompozice tedy získáme čtyři druhy koeficientů, A_1 - aproximační, H_1 - horizontální, V_1 - vertikální a D_1 - diagonální. Další stupně dekompozice se provádějí podobně, rozdíl je jen, jako výše, že vstupními daty pro následující úroveň dekompozice jsou aproximační koeficienty

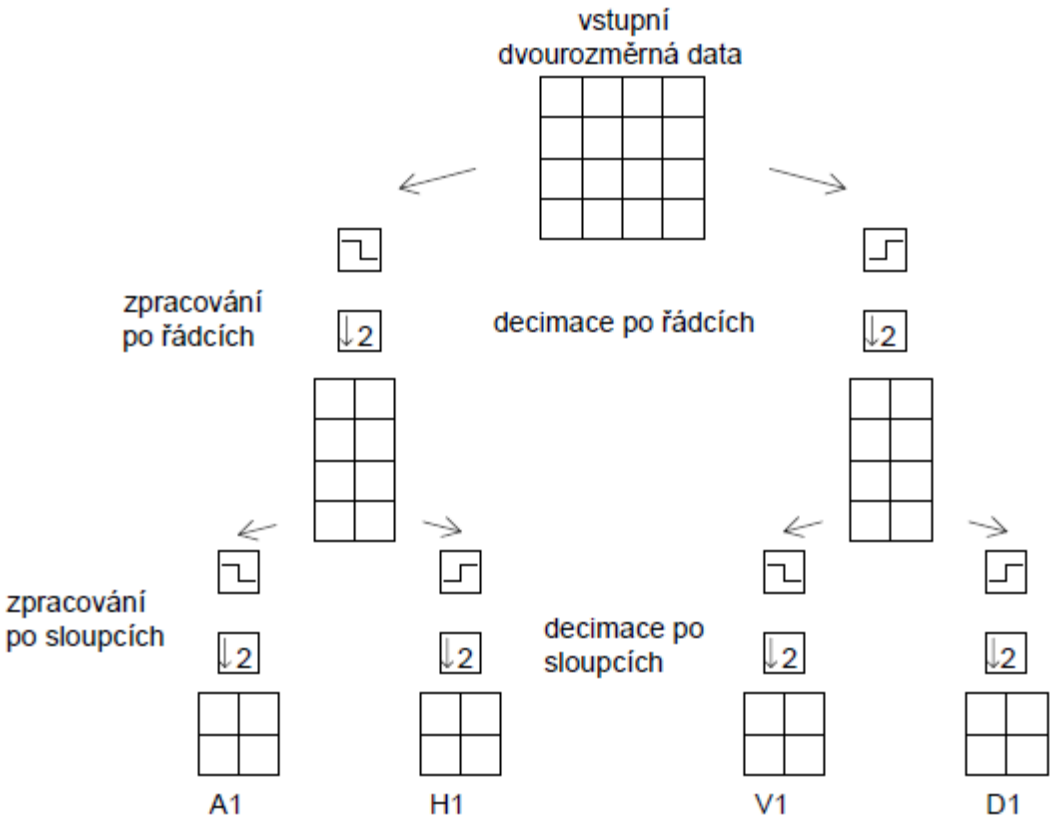


Obr. 3.3: Jeden krok waveletové rekonstrukce. Aproximační a detailní koeficienty jsou nadvzorkovány, filtrovány přes dolní a horní propusti, které jsou inverzní k původním dekompozičním filtrům, a pak sečteny. Pro získání původních aproximačních koeficientů vyšší úrovně je ještě potřeba vyseknout užitečnou část součtu.

aktuálně spočtené úrovně. Horizontální, vertikální i diagonální koeficienty každého kroku samozřejmě uchováváme. Na obr. 3.6 je praktická ukázka dekompozice obrazu do hloubky 2. Vlevo nahoře se nachází signál A_2, který byl dvakrát filtrován filtry typu dolní propust. Je to signálová složka obsahující hodnoty nízkých frekvencí původního signálu, protože byla filtrována filtry typu dolní propust. Nese tudíž největší energii, resp. informaci, protože u všech signálů nesou nízkofrekvenční složky základní informace. Vysokofrekvenční složky naproti tomu nesou detailní informace. U obrazů např. detaily a hrany. Rekonstruovat obraz po waveletové dekompozici je možné inverzním postupem vzhledem k dekompozici. Decimace je nahrazena interpolací, na místo každého původně vynechaného prvku v řádku nebo sloupci se vloží nula. Jako filtry se použijí rekonstrukční DP a HP, které jsou inverzní k těm dekompozičním. Na závěr následuje opět součet a vyseknutí střední části signálu.

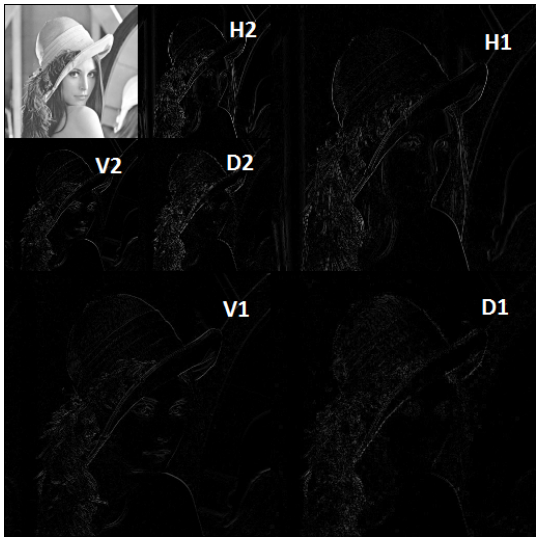
3.8 Okraje signálu

Na závěr této kapitoly je na základě [15] ještě vhodné poznamenat, že základem waveletové transformace, prováděné Mallatovým pyramidovým algoritmem je konvoluce signálu délky n s filtrem délky m. Výsledek konvoluce má pak délku $m+n-1$.



Obr. 3.4: Jeden krok dvourozměrné waveletové dekompozice. Každý vzniklý druh koeficientů má čtvrtinový počet prvků vzhledem ke vstupním datům.

Obr. 3.5: Originální obraz.



Obr. 3.6: Dekompozice do hloubky 2.

Algoritmus je ale navržen pro práci se signálem nekonečné délky. Při použití DTWT tudíž nastává problém, protože pracujeme se signálem konečným. To způsobuje, že okraje signálu nemusí být zpracovány správně. K minimalizaci tohoto problému by bylo nutné znát ještě určitý počet vzorků za okrajem signálu. To ale nebývá možné, proto se signál prodlužuje (extrapoluje). Používané metody jsou doplnění nulami, symetrické prodloužení (zrcadlení), prodloužení polynomem a periodické rozšíření.

4 ŘÍDKÉ REPREZENTACE SIGNÁLŮ

Při návrhu metod pro doplňování pixelů vně obrazu byl vyzkoušen i přístup založený na tzv. řídkých reprezentacích signálů. Mezi implementovanými metodami se tedy nachází i metoda využívající algoritmus Orthogonal Matching Pursuit. Podle [18] jsou základní přístupy pro nalezení řídkých řešení dva. Pro úplnost zmíníme příklady obou.

První skupina metod vychází z ℓ_1-relaxace. Z toho důvodu se algoritmy této skupiny nazývají relaxační. Jsou založeny na předpokladu, že se za určitých podmínek dostaneme k řešení přesnému nebo alespoň relativně blízkému. Jsou to např. BP(Basis Pursuit), modifikovaná LARS (Least Angle Regression, homotopy method) nebo IRLS (Iterative Reweighted Least Squares). Tyto metody jsou zde uvedeny pouze jako příklady z dané skupiny a není s nimi dále pracováno.

Druhou skupinou jsou tzv. „greedy“, neboli hladové algoritmy. Jejich princip spočívá v tom, že v každé iteraci najdou jeden (nebo více) „nejvýznamnějších“ atomů. Důležitý fakt je, že v dalším průběhu algoritmu už vybraný atom nebude zbaven podílu na konečném řešení. Výhodou je nízká složitost, naopak nevýhodou je, že není zaručeno dosažení globálního optima. Z používaných algoritmů uvedeme MP (Matching Pursuit), a v práci použitý OMP (Orthogonal Matching Pursuit).

Samozřejmě se lze setkat s algoritmy hybridními, které využívají výhody obou výše zmíněných skupin.

Pro případné dohledání více informací k problematice řídkých reprezentací signálů doporučujeme začít např. tímto článkem [18], a na něj navazujícími. My se pro účely této práce spokojíme s principem metody MP a na ni navazující OMP, která je využita v praktické části.

4.1 Matching Pursuit

V roce 1993 představili ve [19] Stéphane G. Mallat a Zhifeng Zhang algoritmus, nazvaný Matching Pursuit. Ten rozkládá jakýkoli signál do lineárního rozvoje vlnek, které jsou vybírány z redundantního slovníku funkcí. Tyto vlnky jsou vybírány za účelem co nejlepšího zachycení signálových struktur. Matching pursuits (velmi špatně přeložitelné, proto ponechám původní výraz) jsou obecné procedury k počítání adaptivních reprezentací signálů. Se slovníkem Gaborových funkcí, matching pursuit definuje adaptivní časově-frekvenční transformaci. Distribuce energie signálu je odvozena v časově-frekvenční rovině, která neobsahuje interferenční podmínky. MP izoluje signálové struktury, které jsou koherentní s ohledem na daný slovník. Jejich práce popisuje aplikaci pro extrakci struktur ze zašumělých signálů.

Na základě informací z [19] a [20] uvádím popis metody MP. Matching pursuit je typ numerické techniky, která zahrnuje hledání nejlépe se shodujících projekcí multidimensionálních dat na redundantní slovník $\boldsymbol{D}$. Základní myšlenka je reprezentovat signál f z Hilbertova prostoru $\boldsymbol{H}$, jako váženou sumu funkcí $g_{\gamma n}$ (zvaných atomy) vzatou z $\boldsymbol{D}$.

$$f(t) = \sum_{n=0}^{+\infty} \alpha_n g_{\gamma n}(t) \tag{4.1}$$

kde n indexuje atomy, které byly vybrány a a_n váhovací faktor (amplituda) pro každý atom. Vzhledem k určitému slovníku, bude matching pursuit nejdříve hledat jeden atom, který má největší skalární součin se signálem. Potom odečte příspěvek tohoto atomu a opakuje proces dokud není signál uspokojivě rozložen.

Pro porovnání vezměme v úvahu reprezentaci signálu Fourierovou řadou. Ta může být popsána podobně jako výše s tím, že slovník je vytvořen na základě funkcí sinus. Hlavní nevýhodou Fourierovy analýzy ve zpracování signálů je, že získává pouze globální rysy signálů a neadaptuje se na analyzované signály f. Pokud použijeme extrémně redundantní slovník, můžeme v něm hledat funkce, které nejlépe vystihují signál f. Hledání reprezentací, kde je většina koeficientů v součtu blízká nule (řídké reprezentace) je žádoucí pro signálové kódování a kompresi, ale mohla by být dobře použitelná i pro hledání neznámých hodnot signálu za jeho koncem, pokud jeho známý průběh vyjádříme za použití některé z metod zmíněných výše (např. OMP) a toto vyjádření pak použijeme k odhadu hodnot, které neznáme.

4.1.1 Algoritmus

Hledání nejlepší shody v extrémně velkém slovníku je výpočetně velmi náročné a pro praktickou aplikaci neakceptovatelné. Naštěstí je ale k dispozici řešení představené v [19]. Algoritmus iterativně generuje pro signál f a slovník $\boldsymbol{D}$, tříděný seznam indexů a skalárů, které jsou sub-optimálním řešením problému řídké reprezentace signálu. Zbytek po vypočítání γ_n a a_n je označován R_{n+1}.

Následující pseudo kód pochází z [20].

Algorithm Matching Pursuit
Input: Signal: $f(t)$, dictionary $\boldsymbol{D}$.
Output: List of Coefficients: $(a_n, g_{\gamma n})$.
Initialization:
$R_1 \leftarrow f(t)$;
$n \leftarrow 1$
Repeat:
find $g_{\gamma n} \in \boldsymbol{D}$ with maximum inner product $|\langle R_n, g_{\gamma n} \rangle|$;
$a_n \leftarrow \langle R_n, g_{\gamma n} \rangle$;
$R_{n+1} \leftarrow R_n - a_n g_{\gamma n}$;
$n \leftarrow n + 1$;
Until stop condition (for example: $\|R_n\|$ < threshold)

„$\leftarrow$“ je zkratka pro „změnu na“. Například „$a \leftarrow b$“ znamená, že hodnota „a“ se mění na hodnotu „b“.

4.1.2 Vlastnosti

Algoritmus konverguje pro všechny f v prostoru, který je zahrnut ve slovníku. Pro všechna m je splněna rovnice zachování energie.

$$\|f\|^2 = \sum_{n=0}^{m-1} |a_n|^2 + \|R_m\|^2 \tag{4.2}$$

Chyba $\|R_n\|$ se snižuje monotónně a její pokles je exponenciální.

4.1.3 Užití

Matching pursuit byl použit ke kódování signálů, obrazu a videa, tvarové reprezentaci a rozpoznání, i kódování 3D objektů. Bylo prokázáno [21], že funguje lépe, než kódování založené na DCT pro nízké bitové rychlosti a to jak s ohledem na efektivitu kódování, tak i na kvalitu obrazu.

Hlavní problém s Matching pursuit je výpočetní složitost kodéru. V základní verzi algoritmu musí být v každé iteraci prohledáván velký slovník. Vylepšení zahrnují použití přibližných slovníkových reprezentací a suboptimální způsoby výběru nejlepší shody v každé iteraci.

4.2 Orthogonal Matching Pursuit

Je populární rozšíření Matching Pursuit. Jak napovídá název, je to jeho ortogonální verze. Hlavní rozdíl, kterým se liší od MP je, že po každém kroku jsou aktualizovány všechny až dosud získané koeficienty, a to tak, že je vypočten kolmý průmět

signálu na množinu až dosud vybraných atomů. To může vést k lepším výsledkům, než standardní MP, ale vyžaduje to více výpočtů. Ve skutečnosti tento algoritmus aproximuje řídký problém [20]

$$min_x \|f - D_x\|_2^2 \texttt{ s podmínkou } \|x\|_0 \leq N$$

s $\|x\|_0$ - pseudo normou L_0 (např. počet nenulových prvků x).

Rozšíření jako jsou Multichannel MP a Multichannel OMP umožňují zpracování vícesložkových signálů. Zajímavou variantou je také hybridní algoritmus A* Orthogonal Matching Pursuit, který využívá A* algoritmus prohledávání informačních stromů [22].

5 SROVNÁNÍ KVALITY OBRAZŮ

Tato práce, jak už bylo několikrát zmíněno, se zabývá zpracováním obrazů. Navrhuje několik metod pro doplňování pixelů za vnějším okrajem obrazu. Aby mohlo být dosaženo nějakých závěrů, je potřeba výsledky jednotlivých navržených metod určitým způsobem kvalitativně ohodnotit, resp. srovnat s výsledky ostatních metod. V této kapitole budou proto na základě [23] popsány metody, které takovéto srovnání zajišťují, a vztah mezi nimi. Konkrétně PSNR (Peak Signal to Noise Ratio) a SSIM (Structural Similarity).

5.1 Úvod do problematiky

Každé zpracování obrazu může způsobit ztrátu informace nebo kvality obrazu. Metody pro hodnocení kvality obrazu můžeme rozdělit na objektivní a subjektivní. Subjektivní jsou založeny na lidském úsudku a nevyužívají žádné objektivní kriterium. Objektivní pracují s explicitními numerickými kriterii. Ve vztahu k následujícímu textu budeme předpokládat osmibitový šedotónový obraz.

Mějme referenční obraz f a srovnávací obraz g, oba o rozměru $M \times N$, PSNR mezi f a g je definováno jako

$$PSNR(f,g) = 10\log_{10}(255^2/MSE(f,g)) \tag{5.1}$$

kde

$$MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij}-g_{ij})^2 \tag{5.2}$$

Hodnota PSNR dosahuje nekonečna, pokud je hodnota MSE nula. Z toho plyne, že vyšší hodnota PSNR znamená vyšší kvalitu (v našem případě podobnost) obrazu. Na druhou stranu nízká hodnota PSNR ukazuje na velké rozdíly mezi obrazy. SSIM se považuje za kvalitativní měřítko více odpovídající vnímání lidského visuálního systému (HVS). Namísto použití tradičního sčítání chyb je SSIM navržen tak, že každé obrazové zkreslení je vyjádřeno jako kombinace tří faktorů. Ty jsou ztráta korelace, zkreslení luminance a zkreslení kontrastu. SSIM je definována jako:

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g) \tag{5.3}$$

kde

$$\begin{cases} l(f,g) = \frac{2\mu_f\mu_g+\mathrm{C}_1}{\mu_f^2+\mu_g^2+\mathrm{C}_1} \\ c(f,g) = \frac{2\sigma_f\sigma_g+\mathrm{C}_2}{\sigma_f^2+\sigma_g^2+\mathrm{C}_2} \\ s(f,g) = \frac{\sigma_{fg}+\mathrm{C}_3}{\sigma_f\sigma_g+\mathrm{C3}} \end{cases} . \tag{5.4}$$

První výraz ve 5.4 je luminanční srovnávací funkce, která měří blízkost dvou obrazů. Střední luminance (μ_f a μ_g). Tento faktor je maximální a roven 1, pouze pokud $\mu_f = \mu_g$. Druhý výraz je funkce srovnání kontrastu a měří blízkost kontrastu dvou obrazů. Zde je kontrast měřen pomocí standardní odchylky σ_f a σ_g. Tento výraz je maximální a roven 1, pouze, pokud $\sigma_f = \sigma_g$. Třetí výraz je funkce srovnání struktury, která měří korelační koeficient mezi dvěma obrazy f a g.

Hodnoty indexu SSIM jsou od 0 do 1. Hodnota 0 znamená nulovou korelaci mezi obrazy a 1 znamená, že obrazy jsou totožné. Kladné konstanty $\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3$ jsou použity, abychom se vyhnuli nulovému jmenovateli.

Pro výběr, zda-li k ohodnocení kvality (podobnosti) obrazů použít SSIM nebo PSNR nejsou dána žádná pravidla. Některé studie odhalují, že narozdíl od SSIM, funguje MSE (Mean Squared Error) a tedy i PSNR špatně při rozlišování strukturálního obsahu v obrazech, poněvadž různé druhy degradace obrazu aplikované na stejný obraz mohou poskytovat stejnou hodnotu MSE. Jiné studie ukázaly, že MSE, a v důsledku toho PSNR má nejlepší výkon v hodnocení kvality zašumělých obrazů. V následující kapitole bude odvozen vztah mezi SSIM a PSNR, což umožní lépe porozumět jejich rozdílům a podobnosti pro použití při běžných znehodnoceních, jako jsou Gaussovské rozmazání, aditivní Gaussovský šum, JPEG komprese nebo právě degradace vzniklé použitím extrapolačních metod.

5.2 Vztah mezi PSNR a SSIM

K ustanovení vztahu mezi SSIM a PSNR bude nejdříve odvozen vztah mezi SSIM a MSE. Potom bude tento vztah použit ke svázání SSIM s PSNR. MSE v rovnici 5.2 může být přepsána jako:

$$MSE = \sigma_f^2 + \sigma_g^2 - 2\sigma_{fg} + (\mu_f - \mu_g)^2 \tag{5.5}$$

kde σ_f^2 a σ_g^2 jsou odchylky obrazů f a g, a σ_{fg} je kovariance mezi f a g:

$$\sigma_f^2 = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - \mu_f)^2,\ \sigma_{fg} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - \mu_f)(g_{ij} - \mu_g). \tag{5.6}$$

SSIM definována v 5.3 může být přepsána jako:

$$\frac{1}{SSIM} = \frac{255^2 \times \alpha(f,g) \times e^{-PSNR \times \ln(10)/10} + \beta(f,g)}{l(f,g)s(f,g)} \tag{5.7}$$

kde

$$\alpha(f,g) = \frac{1}{2\sigma_f\sigma_g + C_2},\ \beta(f,g) = \frac{2\sigma_{fg} - (\mu_f - \mu_g)^2 + C_2}{2\sigma_f\sigma_g + C_2},\ s(f,g) = \frac{\sigma_{fg} + C_3}{\sigma_f\sigma_g + C_3} \tag{5.8}$$

Předpokládejme nyní, že $C_2 \ll \sigma_f, \sigma_g$ a $C_3 \ll \sigma_f, \sigma_g$. Tento předpoklad jsme učinili proto, abychom vynulovali vliv konstant, které se objevují ve vztahu pro SSIM. Tyto konstanty byly totiž zavedeny, abychom se zbavili nulového jmenovatele. Tudíž, v případě nenulových hodnot odchylek, mohou být konstanty zahozeny. Nenulové hodnoty odchylek se nacházejí v obrazech, ve kterých alespoň jeden pixel má hodnotu úrovně šedé rozdílnou od ostatních. V tomto případě lze z 5.7 a 5.8 odvodit, že:

$$PSNR = 10\log_{10}\left[\frac{2\sigma_{fg}(l(f,g) - SSIM)}{255^2\, SSIM} + \left(\frac{\mu_f - \mu_g}{255}\right)^2\right] \tag{5.9}$$

Vztah v 5.9 je obecný a může být použit pro všechny druhy degradace obrazu. Vztah může být dále zjednodušen v případě konkrétní obrazové degradace.

5.3 Shrnutí

Po analyzování vztahů pro výpočet PSNR a SSIM bylo zjištěno, že mezi těmito kvalitativními měřítky existuje analytický vztah, který funguje pro běžné degradace obrazu, jako Gaussovské rozmazání, aditivní Gaussovský šum, nebo JPEG komprese. Z experimentálních výsledků provedených ve [23] vyplývá, že PSNR je více citlivý na aditivní Gaussovský šum než SSIM, zatímco opak platí pro JPEG kompresi. Obě metody mají zhruba stejnou citlivost na Gaussovské rozmazání. Ve všech případech bylo pozorováno, že PSNR a SSIM jsou více citlivé na aditivní Gaussovský šum než na Gaussovské rozmazání a JPEG kompresi. Jako poslední závěr stojí za zmínku, že hodnoty PSNR mohou být předpovězeny z SSIM a naopak. Pro naše konkrétní účely bude použito pro srovnání původních a zpracovaných obrazů obou metod, neboť degradace a rozdíly, které vzniknou, mohou být různého charakteru. Proto nelze jednoduše určit, které z metod by měl být přikládán větší význam.

6 PRAKTICKÉ VÝSLEDKY

6.1 Software

Před začátkem práce bylo důležitým krokem rozhodnutí, ve kterém programovacím jazyce implementovat algoritmy řešící zadanou problematiku. Volba padla na MATLAB, protože tento systém je velmi komplexní, a ve srovnání s jinými programovacími jazyky, i poměrně jednoduchý. Dovolíme si jej zde blíže představit. Popisován je i z toho důvodu, že někteří budoucí čtenáři této práce nemusí mít s programem zkušenosti a tudíž pro ně bude nezbytné pochopit alespoň základní principy fungování. Informace níže pocházejí z části z [8], kde jsou přehledně shrnuty a z části z webové stránky společnosti MathWorks [9]. Pro ty, kteří by chtěli začít vyvíjet v prostředí MATLAB je vhodnou literaturou [10], [11] a [12].

6.1.1 Stručná charakteristika MATLABU

MATLAB pochází ze slovního spojení MATrix LABoratory, což znamená maticová laboratoř. Matice jsou totiž klíčovou datovou strukturou pro výpočty v něm. Je to programové prostředí a skriptovací programovací jazyk. Program MATLAB je vyvíjen společností MathWorks a zatím poslední verze je R2013a. MATLAB je k dispozici pro operační systémy Linux (32-bit, 64-bit), Windows (32-bit, 64-bit) a Mac OS X (64-bit). V současnosti existují i verze pro iOS a Android, které však pracují na principu Cloud Computing a připojují se buď na domácí PC uživatele nebo na cloud server splečnosti MathWorks, která tento software vyvíjí.

MATLAB umožňuje počítání s maticemi, vykreslování 2D i 3D grafů funkcí, implementaci algoritmů, počítačovou simulaci, analýzu a prezentaci dat i vytváření aplikací včetně uživatelského rozhraní, tzv. GUI. Původně byl jazyk určen pro matematické účely, ale časem byl upraven, byly přidány nové funkce a rozšíření, rozrostl se různými směry a dnes je využitelný v široké paletě aplikací. MATLAB má mnoho uživatelů především z řad vědeckotechnických pracovníků, studentů a zaměstnanců vysokých škol. Komerčně se systém samozřejmě také používá, ale náklady na jeho pořízení jsou značné a závisí na různých volitelných součástech, např. na počtu toolboxů. MATLAB je využíván pro vědecké a výzkumné účely a to jak v soukromém sektoru, tak i v akademických řadách. Hlavní oblastí využití jsou technické obory a ekonomie.

6.1.2 Historie

MATLAB vytvořil na konci sedmdesátých let profesor Cleve Moler, který v té době působil na Univerzitě v Novém Mexiku na katedře informačních technologií. Navrhl MATLAB, aby jeho studenti mohli využívat LINPACK a EISPACK bez nutnosti se učit programovací jazyk Fortran, který se mnoho let využíval jako hlavní jazyk pro matematické výpočty. Programování ve Fortranu bylo složité a skrývalo mnoho nástrah při řešení matematických výpočtů. MATLAB se velice rychle rozšířil i na další univerzity a získal si veliké množství uživatelů a fanoušků. Široké využití našel především v aplikované matematice. V roce 1983 když byl Cleve Moler na návštěvě na Stanford University se o MATLAB začal zajímat Jack Little, který v softwaru uviděl ekonomický potenciál. Do té doby byl MATLAB zdarma. Jack Little přepsal MATLAB do jazyka C, přidal některé další funkce a knihovny a v roce 1984 založili Little, Moller a Steve Bergert společnost MathWorks, která dále pokračovala ve vývoji a nabídla produkt na trh.

První verze pro PC XT [1] byla vydána kolem roku 1985. Základním problémem byl nedostatek paměti a z toho plynoucí omezení na maximální velikost matic. Po příchodu PC AT [2] rychle následovala verze MATLABu pro tento počítač. Zde byla maximální velikost matice omezena fyzickou velikostí paměti (max 16 MiB). Vzhledem k tehdejším cenám paměti však nebylo osazení větší paměti běžné. Velké obliby doznala verze MATLAB 386, která byla určena pro PC s 32-bitovým procesorem Intel 80386, která využívala virtuální paměť. Program pracující s virtuální pamětí umožňuje využívat více operační paměti, než je dostupná fyzická paměť RAM, protože dochází k odkládání dat na pevný disk počítače. Tato skutečnost sice vede ke zpomalení výpočtů, ale je alespoň možné výpočty na velkých maticích vůbec provést.

6.1.3 Vlastnosti

Programovací jazyk Matlab je integrované prostředí, které je určené pro vědecko-technické účely, simulace, paralelní výpočty apod. Zahrnuje výpočty, vizualizaci a programování do uživatelsky ovladatelného prostředí. Typické oblasti použití jsou:

- inženýrské výpočty,
- tvorba algoritmů,
- modelování a simulace,
- analýza dat,
- vědecká a inženýrská grafika,

[1] IBM Personal Computer XT, neboli IBM PC/XT je zkratka z X-tended Technology. Jinak také IBM 5160

[2] IBM Personal Computer AT, neboli IBM 5170 je označení pro třetí generaci IBM PC kompatibilních počítačů

- tvorba aplikací (včetně grafického rozhraní).

Mezi základní vlastnosti lze zahrnout, že veškeré objekty v MATLABu jsou považovány za matice. Tyto však mohou být nejen čísla, proměnné, ale i složitější struktury, jako například obrázky. Výkonnost MATLABu je rozšiřována díky navazujícímu softwaru, který tvoří především tzv. toolboxy (nástrojové sady), orientované zpravidla na daný problém nebo uživatelem sestavené programy, tzv. m-files (m-soubory).

6.1.4 Proměnné v Matlabu

MATLAB má slabou dynamickou typovou kontrolu. To znamená, že proměnné v MATLABu nemají po deklaraci určený datový typ a mění datový typ během své existence. Je možné do jedné proměnné uložit datový typ integer a následně v kódu do té samé proměnné uložit textový řetězec, kterým přepíšeme původní hodnotu. Dynamické typování je pružnější a mnohdy pohodlnější pro programátora, ovšem je daleko náchylnější ke vzniku chyb. Programátor si typovou kontrolu musí hlídat sám a některé takto vzniklé chyby se projeví až daleko od místa svého vzniku a jsou tak jen těžko odhalitelné.

Proměnné v Matlabu nevyžadují deklaraci, vzniknou prvním přiřazením hodnoty. Čtení z neexistující proměnné vyvolá chybu. Změny typu a velikosti proměnných probíhají automaticky.

Základním druhem proměnné je matice. Každá matice může v MATLABu obsahovat:

- čísla v různých formátech (celočíselná, s plovoucí desetinnou čárkou, komplexní),
- znaky (s vektorem znaků je zacházeno jako s řetězcem),
- struktury, které mohou obsahovat další matice nebo struktury,
- symbolické proměnné nebo speciální typy (např. přenosová funkce 'tf').

Matice může mít prakticky libovolný počet rozměrů, v každém rozměru jsou prvky označeny celými kladnými čísly od 1 do n. Matice o rozměrech 1×1 je označována jako skalár, matice $1 \times n$ nebo $m \times 1$ je označována jako vektor. Všechny prvky jedné matice musí být stejného typu ('double', 'int', 'logical', 'struct', 'sym', ...).

Omezení matic na stejný typ ve všech prvcích odstraňuje později zavedení proměnné „cell“, k jejímž prvkům se přistupuje pomocí složených závorek. Jednotlivé prvky „cell“ mohou mít libovolný obsah.

6.1.5 Co je ještě dobré vědět o MATLABu

Za zmínku stojí program Simulink, který využívá Matlab a jeho funkce k simulaci dynamických systémů. Proměnné se mezi MATLABem a Simulinkem sdílejí přes „workspace“, což je volně přeloženo „pracovní prostor“. Zde se zobrazují názvy, velikosti a datové typy všech momentálně používaných proměnných.

Jako poslední zmíním, že MATLAB podporuje objektově orientované programování. V MATLABu můžete vytvářet třídy, používat dědičnost, nastavovat události a posluchače pro pozorování objektů nebo využívat návrhových vzorů používaných i v jiných objektově orientovaných jazycích. OOP významně zjednodušuje a zpřehledňuje tvorbu složitějších aplikací.

6.2 Praktické řešení

V rámci praktické části řešení projektu jsem implementoval osm metod pro extrapolaci obrazu. Jsou to:

- **Metoda OMP**, využívající algoritmus Orthogonal Matching Pursuit a řídkou reprezentaci signálů.
- **Metoda aproximace křivkou**, extrapolující obraz po řádcích s využitím funkce `polyfit` vestavěné v MATLABu. Funkcí `polyval` se pak vypočte polynom na nových souřadnicích za okrajem obrazu.
- **Metoda aproximace plochou**, počítá extrapolaci podobným způsobem, ale nevychází z průběhu řádku, ale plochy na kraji obrazu, základní funkce se jmenuje `polyfit2d` a jejím autorem je Perry Stout. Funkce je volně šiřitelná. V této metodě se používá podobná funkce, jako `polyval`, a sice `polyval2d` od stejného autora.
- **Metoda dvojité aproximace**, extrapoluje obraz po řádcích s použitím dvou polynomů různého řádu.
- **Metoda mediánového filtru**, nejdříve vyhladí část obrazu použitou pro výpočet aproximace mediánovým filtrem a až pak se snaží průběh aproximovat.
- **Metoda s detekcí hran** detekuje hrany v části obrazu, která je použitá pro výpočet aproximačního polynomu a podle výskytu hran upraví počet pixelů, který se použije pro výpočet. Metoda se tak snaží eliminovat postupně se zvyšující nebo snižující hodnoty jasu, které mají za následek to, že extrapolační polynom se rychle dostává mimo zobrazitelný rozsah hodnot.
- **Metoda waveletového rozkladu** se podobně jako metoda mediánového filtru snaží ubrat obrazu detailní informace tím, že obraz rozloží na méně detailní. Tím by mělo dojít k eliminaci vysokofrekvenčních složek.

- **Metoda zrcadlení** symetricky prodlouží obraz kolem osy, kterou tvoří okraj obrazu. Metodu je možné uplatnit pouze na speciálních typech obrazu.

Na začátku každého ze skriptů, implementujících dané metody se vstupní obraz otestuje, zda-li je barevný nebo šedotónový. Skripty jsou napsány pro přehlednost pro šedotónové obrázky, tudíž, je-li obrázek barevný, převede se na šedotónový.

Dále je z obrazu odejmut kontrolní úsek. Jedná se o matici stejného rozměru, jaká bude dále počítána. Tento krok slouží k tomu, aby bylo možné porovnat vypočítaný úsek se skutečnými daty a ohodnotit tak kvalitu metody. Hodnocením kvality se budeme zabývat později.

6.2.1 Metoda OMP

Vstupními parametry pro tuto metodu jsou `smer`, který definuje, na kterou stranu obrazu se budou pixely doplňovat a `ppp`, který určuje, o kolik se obraz do zadaného směru prodlouží. Dále `wavelet`, který definuje wavelet použitý pro výpočet báze a `depth`, určující hloubku waveletové dekompozice.

```
%% vstupni parametry
smer = 'r'; % l,r,u,d pro zmenu smeru extrapolace obrazku
ppp = 32; % pocet pixelu, o kolik se bude obrazek prodluzovat
wavelet = 'db3'; % pouzity wavelet
depth = 8; % hloubka waveletove dekompozice
```

Metoda OMP pro každý sloupec obrazu vezme vzorky (pixely), které zná a pro každý jednotlivý pixel od něj odečte průměrnou hodnotu pixelu vypočtenou pro celý sloupec. Tím se dostane s hodnotami do oblasti okolo nuly. Pak se vypočítají bázové vektory DWT. Definují se kritéria pro zastavení vykonávání pro funkci `omp`. Jsou to maximální počet iterací a maximální chyba. Autory funkce `omp`, realizující algoritmus OMP jsou Pavel Rajmic, Jan Špiřík, Marie Danková a Václav Mach, tato funkce využívá ke své funkci některé další skripty, jejichž autory jsou Zdeněk Průša a Pavel Rajmic. Funkce `omp` vrátí souřadnice řídkého řešení, nalezené pomocí algoritmu OMP a signál se rekonstruuje jejich vynásobením s vypočítanou bází. Takto rekonstruovaný signál již obsahuje i informaci o hodnotách pixelů doplněných za okrajem signálu (původního sloupcového vektoru). Nakonec se ke každému pixelu přičte původně odečtený průměr a hodnoty vektoru se tak vrátí do původního rozsahu. Všechny sloupce se postupně sloučí do výsledné matice obrazu a obraz se vykreslí.

6.2.2 Metoda aproximace křivkou a metoda aproximace plochou

Před spuštěním skriptu je potřeba nastavit počáteční parametry. Jsou zde patrné z ukázky zdrojového kódu, jejich význam je vysvětlen dále.

```
%% nastaveni parametru
smer = 'r'; % l,r,u,d pro zmenu smeru extrapolace obrazku
metoda = 'ext_krivkou'; % ext_krivkou, ext_plochou
n = 1; % stupen polynomu
m = 3; % stupen polynomu ve druhem smeru pro volbu 'ext_plochou'
ppp = 32; % pocet pixelu prodlouzeni - o kolik se obraz prodlouzi
pocet_pixelu_pro_vypocet = 32; %pocet pixelu pro vypocet polynomu
zobraz_radky = []; %vektor radku, ktere chceme zobrazit graficky
```

Volba `smer` nastaví, na kterou stranu se bude obraz prodlužovat. Metoda aproximace křivkou a metoda aproximace plochou jsou spouštěny ze stejného skriptu, proto je potřeba definovat, která metoda bude použita. K tomuto účelu slouží proměnná `metoda`. Účel `n` a `m` je zřejmý, slouží k nastavení stupně polynomu, `n` pro křivku nebo první směr plochy, `m` pro druhý směr plochy v případě použití metody aproximace plochou. Proměnná `ppp` definuje, kolik pixelů se bude dopočítávat, zatímco `pocet_pixelu_pro_vypocet` určuje, kolik hodnot z předchozího průběhu (pixelů v řádku obrazu) se bude používat pro výpočet koeficientů polynomu daného řádu. Pokud by byl totiž použitý úsek signálu velký, v krajním případě celý řádek obrazu, nepopisovaly by vypočtené hodnoty dobře úsek blízký okraji, tedy sousedící s neznámou oblastí. Uplatnily by se zde totiž veškeré hodnoty jasu, vyskytující se na daném řádku. V případě použití metody pro aproximaci plochou musejí být `ppp` a `pocet_pixelu_pro_vypocet` nastaveny na stejnou hodnotu, funkce `polyfit2d` totiž pracuje pouze se čtvercovou plochou. Poslední volbou je `zobraz_radky`, do které můžeme zapsat čísla řádků obrazu, jejichž průběh hodnot jasu bychom si rádi nechali vykreslit do grafu. Implementovaná funkce vykresluje i průběh polynomu, vypočítaného na daném řádku, a to v aproximační i extrapolované části.

Princip metody aproximace křivkou je takový, že se funkcí `polyfit` vypočítá aproximační polynom zadaného řádu pro určený počet posledních pixelů řádku. Potom se vypočítá hodnota pixelů za hranicí obrazu podle tohoto polynomu. Původní a vypočtená část se sloučí do jediného obrazu, který je výsledkem.

Metoda aproximace plochou nevyužívá k extrapolaci jednorozměrnou funkci `polyfit`, nýbrž dvourozměrnou funkci `polyfit2d`. Tato funkce narozdíl od předchozí nepočítá aproximační křivku, ale aproximuje pomocí dvourozměrné funkce, tj. plochy. Na koncové části obrazu se zvolí velikost plochy, která bude použita pro aproximaci. Na ní se vypočítá matice koeficientů polynomu, podobně jako v první funkci vektor koeficientů polynomu. Tato matice se pak aplikuje na souřadnice za

hranicí obrazu a vypočtou se extrapolované hodnoty v těchto souřadnicích. Zapíše se ale jen první řádek. Pak se algoritmus posune o jeden řádek v obrazu dolů a použije se nová aproximační plocha a vypočte nová extrapolovaná plocha. N-tá extrapolovaná plocha se aritmeticky zprůměruje s N-1 extrapolovanou plochou v souřadnicích, kde se překrývají a první řádek tohoto průměru se zapíše do obrazu. Takto se zachází se všemi řádky obrazu. Výsledné extrapolované hodnoty jsou tudíž průměrem dvou extrapolací plochou. Očividná nevýhoda tohoto přístupu je, že první a několik posledních řádků (podle rozměru plochy) takto zprůměrovat nelze, protože posunutí aproximační plochy byť jen z části mimo obraz by zkreslilo její průběh. Resp. MATLAB by zahlásil chybu, protože hodnoty mimo obrazovou matici nejsou definovány. Původní a vypočtená část se opět sloučí do jediného obrazu, který je výsledkem.

6.2.3 Metoda dvojité aproximace

Začneme opět nastavením parametrů. Většina je stejná, jako v předchozím případě. Jak napovídá název metody, budeme používat dvou různých stupňů polynomu. Nižší stupeň udává `n1`, vyšší pak `n2`.

```
%% nastaveni parametru
smer = 'r'; % l,r,u,d pro zmenu smeru extrapolace obrazku
ppp = 32; % pocet pixelu prodlouzeni - o kolik se obraz prodlouzi
pocet_pixelu_pro_vypocet = 32; %pocet pixelu pro vypocet polynomu
n1 = 1; % rad pouziteho aproximacniho polynomu NIZSIHO stupne
n2 = 2; % rad pouziteho aproximacniho polynomu VYSSIHO stupne
```

Tato metoda vychází z předpokladu, že v bodech vzdálených od uzlových bodů nabývá chyba při výpočtu polynomů velkých hodnot. Pokud se snažíme o extrapolaci signálu, tj. o odhad nebo výpočet hodnot signálu v místě, kde je neznáme za využití hodnot v místech, které jsou nám známy, tento předpoklad se uplatní. Pro případ, kdy se snažíme aproximovat signál polynomem vysokého řádu budou vypočtené hodnoty těsně za známými hodnotami poměrně přesné. Nemusí to být pravidlo, avšak velmi pravděpodobně budou přesnější, než hodnoty v místech vzdálenějších. Naproti tomu použitím polynomu nízkého řádu se v bližších bodech sice nedostaneme tak blízko skutečným hodnotám, jako při použití vysokého řádu, ale v bodech vzdálenějších by chyba neměla růst tak rychle. Přístup této metody je proto takový, že se vypočítá extrapolace za použití nízkého i vyššího řádu. Hodnoty se potom váhují a sečtou. A to tak, že směrem od okraje obrazu dál klesá vliv hodnot získaných za pomoci polynomu vyššího řádu a stoupá vliv hodnot vypočtených polynomem nízkého řádu. Matice vypočtených hodnot se sloučí s nezpracovávanou částí obrazu a výsledek se zobrazí.

6.2.4 Metoda mediánového filtru

Všechny zde použité vstupní parametry byly již vysvětleny výše.

```
%% nastaveni parametru
smer = 'r'; % l,r,u,d pro zmenu smeru extrapolace obrazku
ppp = 32; % pocet pixelu prodlouzeni - o kolik se obraz prodlouzi
n = 1; % rad pouziteho aproximacniho polynomu
pocet_pixelu_pro_vypocet = 32; %pocet pixelu pro vypocet polynomu
```

Na začátku metody je definována oblast, ze které se budou počítat aproximační polynomy. Protože v rámci této metody chceme vyhladit obraz, aby oblast, na které se počítá aproximace, neobsahovala hodnoty nadměrně se odchylující od hodnot okolí, vyfiltruje se tato oblast mediánovým 2D filtrem. Na základě takto upravené části matice obrazu doplníme zadaný počet pixelů za okraj obrazu. Výsledek se vykreslí.

6.2.5 Metoda s detekcí hran

Všechny zde použité vstupní parametry mají stejný význam jako u předchozí metody mediánového filtru. Výjimkou je `pocet_pixelu_pro_vypocet`, který zde neudává přesný počet použitých pixelů, ale maximální mez. Detailněji to bude vysvětleno dále.

```
%% nastaveni parametru
smer = 'r'; % l,r,u,d pro zmenu smeru extrapolace obrazku
ppp = 32; % pocet pixelu doplnovanych na konec obrazu
n = 1; % rad pouziteho aproximacniho polynomu
pocet_pixelu_pro_vypocet = 32; %pocet pixelu pro vypocet polynomu
```

Tato metoda se snaží zkoumat vliv počtu pixelů, které se použijí pro výpočet aproximačního polynomu. Úsek definovaný proměnnou `pocet_pixelu_pro_vypocet` totiž může obsahovat hrany, což znamená skokové změny jasových hodnot. Metoda tudíž nejdříve detekuje hrany v části obrazu určené pro aproximování, a na základě toho upraví počet pixelů, které se skutečně použijí pro výpočty. Tedy vezme jen hodnoty mezi poslední detekovanou hranou obrazu a jeho okrajem. Pokud hrana dosahuje až na okraj obrazu, dojde k prodloužení konstantou. Příklad je na obr. 6.1.

6.2.6 Metoda waveletového rozkladu

Význam vstupních parametrů je patrný z ukázky zdrojového kódu, proto jen stručně. Volba `smer` určuje, kterým směrem se bude doplňování pixelů provádět, `DL` udává

Obr. 6.1: Znázornění principu metody s detekcí hran. Levá část je originální obraz, směrem doprava jsou zobrazeny detekované hrany a nakonec doplněné pixely. Výpočty se provádějí na původním obrazu. Černobílý úsek je pouze ilustrativní, pro zobrazení detekovaných hran.

hloubku dekompozice, maximální je pět. Parametr `vlnka` určuje typ použitého waveletu, resp. dekompozičních a rekonstrukčních filtrů. Ostatní parametry mají stejný význam jako výše.

```
%% nastaveni parametru
smer = 'r'; % l,r,u,d pro zmenu smeru extrapolace obrazku
DL = 1; % Decomposition Level (bezne znaceno jako J), max = 5
vlnka = 'db1';
ppp = 32; % mocniny 2... pocet pixelu doplnovanych na konec obrazu
n = 1; % rad pouziteho aproximacniho polynomu
pocet_pixelu_pro_vypocet = 32; %pocet pixelu pro vypocet polynomu
```

Algoritmus této metody funguje tak, že vstupní obraz rozloží pomocí waveletové dekompozice na aproximační a detailní koeficienty až do zadané úrovně. Na matici aproximačních koeficientů provede výpočet pixelů za hranicí obrazu pomocí funkcí

`polyfit` a `polyval` při takto sníženém prostorovém rozlišení. Podobný postup je aplikován zvlášť na matice detailních koeficientů. Rozšířené matice jsou zpětně rekonstruovány a zobrazeny v podobě zpracovaného obrazu.

6.2.7 Metoda zrcadlení

U této metody se nastavuje pouze směr, ve kterém bude metoda aplikována a počet pixelů, které budou doplňeny za okraj obrazu.

```
%% nastaveni parametru
smer = 'r'; % l,r,u,d pro zmenu smeru extrapolace obrazku
ppp = 32; % pocet pixelu doplnovanych na konec obrazu
```

Fungování metody je zřejmé z názvu. Vezme se zadaný počet pixelů, který má být doplněn za okraj obrazu. Na vnitřním okraji zdrojového obrazu se vysekne matice odpovídající rozměru obrazu v extrapolovaném směru krát zadaný počet pixelů, který je potřeba doplnit. Matice se otočí kolem osy symetrie, kterou tvoří okraj obrazu a přidá se do původního obrazu. Na obr. 6.2 můžeme vidět výstup metody použité na vhodný typ obrazu.

6.3 Zpracování výsledků a porovnání metod

Pro účely testování algoritmů jsem vytvořil testovací a zobrazovací skripty. Testovací skript `test` při svém spuštění zavolá pro zadaný obrázek všechny implementované metody, vypočítá pro každou z nich hodnotu PSNR a SSIM, které jednak vypíše na obrazovku a jednak uloží do vektorů, aby se daly rovnou použít ve zobrazovacím skriptu. Zobrazovací skript `grafy` má na pevno nadefinovány vektory s hodnotami PSNR a SSIM získanými pro testovací obrázky. Umožňuje však jednoduše načíst vektory z předchozího skriptu a vykreslit grafické srovnání PSNR a SSIM pro jednotlivé metody aplikované na zadaný obraz. Výstupy srovnání v podobě tabulek, grafů a příkladů zpracovaných obrazů lze nalézt v příloze.

Autorem funkce použité pro výpočet SSIM je Zhou Wang, funkce je použitelná pro vzdělávací a výzkumné účely.

Výchozí parametry, se kterými byly skripty spouštěny v testovacím procesu, ze kterého jsou zde zpracovány výsledky, jsou patrné pro každou metodu z ukázek zdrojového kódu.

Obr. 6.2: Obrázek po zpracování metodou zrcadlení. Vidíme, že pro některé typy obrazů dává metoda dobré výsledky.

7 ZÁVĚR

V práci byly popsány některé metody pro inpainting obrazu pomocí počítače. Seznámili jsme se s možnostmi programovacího jazyka a prostředí MATLAB. V něm bylo implementováno několik metod pro řešení zadané problematiky. V průběhu realizace se ukázalo, že použití polynomů vysokých řádů zapříčiňuje rychlý růst chyby ve větších vzdálenostech od okraje obrazu. Na základě toho byla vytvořena metoda dvojité aproximace, která pro vzdálenější hodnoty používá polynomů nízkých řádů.

Další přístup se opíral o myšlenku potlačit detaily v obraze. Zde se naskytla možnost práce s obrazem v jiné reprezentaci, tzn. transformace. Fourierova transformace je nepoužitelná, neboť zachycuje spíše globální charakter signálu a v důsledku toho by byl obraz velmi degradován, např. rozmazán. Volba tedy padla na waveletovou transformaci, která ale dává velmi podobné výsledky, jako metoda mediánového filtru. Tato metoda před výpočtem aproximace vyhlazuje obraz, aby oblast, na které se počítá aproximace, neobsahovala hodnoty nadměrně se odchylující od hodnot okolí. Metoda mediánového filtru je navíc implementačně mnohem jednodušší, než metoda waveletového rozkladu.

Detekce hran a od ní se odvíjející dynamická změna délky signálu, která se použije pro aproximaci, ve srovnání s ostatními metodami lepší výsledky nepřinesla.

Jednoduchá, ale zajímavá je metoda zrcadlení, která působí kvalitativně dobře, ale jen pro obrazy, kde není ozrcadlení dané části výrazně patrné.

Práce dále obsahuje popis metod pro srovnávání kvality obrazů a tyto metody jsou také implementovány. Jedná se o Peak Signal to Noise Ratio (PSNR), který vzájemně hodnotí dva obrazy na základě výpočtu Mean Squared Error (MSE), a Structural Similarity (SSIM), jehož hodnota ve výsledku více odráží subjektivní dojem člověka o kvalitě srovnávaných obrazů.

Výsledky jsou zpracovány ve formě tabulek C.1 a C.2. Všechny metody jsou testovány na sedmi obrazech různého charakteru. Dále jsou výsledky srovnávacích metod graficky zobrazeny, a to pro všechny metody aplikované vždy na jeden obraz. Tyto grafy lze nalézt v příloze. Jedná se o grafy D.1 až D.7. Pro vybraný testovací obraz jsou v příloze na ukázku také výsledky všech metod v podobě zpracovaného obrazu B.1 až B.8.

Výrazně lepších výsledků oproti ostatním metodám dosáhla podle PSNR metoda zrcadlení, a to pro obrazy krajina, luxor a mraky, a metoda OMP pro obraz budova a tráva. Podle SSIM dosáhly všechny metody vysokého indexu pro obrazy geometrické tvary a mraky, a zároveň všechny dosáhly pouze nízkých hodnot SSIM pro obraz tráva. Z toho plyne, že použité přístupy dosahují lepších výsledků na obrazech, kde se hodnoty sousedních pixelů mění pozvolna, naopak horších výsledků pro obrazy s výraznými hranami a rychle se měnícími hodnotami sousedních pixelů.

LITERATURA

[1] SHEN, Jianhong. *Inpainting and the Fundamental Problem of Image Processing* [online]. 2002, s. 6 [cit. 30. 11. 2012]. Dostupné z URL: <http://www.math.ucla.edu/~imagers/htmls/internalreport/ShenSIAM.pdf>.

[2] MUTHUKUMAR, S, N. KRISHNAN, P. PASUPATHI and S. DEEPA. *Article:Analysis of Image Inpainting Techniques with Exemplar, Poisson, Successive Elimination and 8 Pixel Neighborhood Methods* [online]. International Journal of Computer Applications 9(11):15–18, November 2010. Published By Foundation of Computer Science. [cit. 07. 12. 2012]. Dostupné z URL: <http://www.ijcaonline.org/volume9/number11/pxc3871928.pdf>.

[3] XU, Jing, Daming FENG, Jian WU and Zhiming CUI. *An Image Inpainting Technique Based on 8-Neighborhood Fast Sweeping Method* [online]. WRI International Conference on Communications and Mobile Computing. 2009, vol. 3, s. 626-630 [cit. 08. 12. 2012]. Dostupné z URL: <http://origin-www.computer.org/plugins/dl/pdf/proceedings/cmc/2009/3501/03/3501c626.pdf>.

[4] SHEN, Jianhong, Sung Ha KANG and Tony F. CHAN. *Euler's Elastica and Curvature-Based Inpaintings. SIAM Journal on Applied Mathematics* [online]. 2003, vol. 63, issue 2, s. 564-592 [cit. 20. 05. 2013]. DOI: 10.1137/S0036139901390088. Dostupné z URL: <http://epubs.siam.org/doi/abs/10.1137/S0036139901390088>.

[5] IGEHY, Homan and Lucas PEREIRA. *Image Replacement through Texture Synthesis. Proceedings of the 1997 IEEE International Conference on Image Processing* [online]. 1997, s. 4 [cit. 20. 05. 2013]. Dostupné z URL: <https://graphics.stanford.edu/papers/texture_replace/texture_replace.pdf>.

[6] CHAN, Tony F. and Jianhong SHEN. *Morphologically Invariant PDE Inpaintings* [online]. s. 14 [cit. 20. 05. 2013]. Dostupné z URL: <http://conservancy.umn.edu/bitstream/3602/1/1770.pdf>.

[7] CHAN, Tony F. and Jianhong SHEN. *Non-Texture Inpainting by Curvature-Driven Diffusion (CDD)* [online]. 2000, s. 16 [cit. 20. 05. 2013]. Dostupné z URL: <ftp://ftp.math.ucla.edu/pub/camreport/cam00-35.pdf>.

[8] *MATLAB* [online]. 29. 11. 2012, [cit. 01. 12. 2012]. Dostupné z URL: <http://cs.wikipedia.org/wiki/MATLAB>.

[9] THE MATHWORKS, Inc. *MATLAB: The Language of Technical Computing* [online]. [cit. 01. 12. 2012]. Dostupné z URL: <http://www.mathworks.com/products/matlab/>.

[10] ZAPLATÍLEK, Karel. *MATLAB pro začátečníky 1. díl.* 2. vyd. Praha: BEN - technická literatura, 2005, s. 151. ISBN 80-730-0175-6.

[11] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. *MATLAB: tvorba uživatelských aplikací 2. díl.* 1. vyd. Praha: BEN - technická literatura, 2004, s. 215. ISBN 80-730-0133-0.

[12] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. *MATLAB: začínáme se signály 3. díl.* 1. vyd. Praha: BEN - technická literatura, 2006, s. 271. ISBN 80-730-0200-0.

[13] RŮŽIČKOVÁ, Irena a Rudolf HLAVIČKA. *Numerické metody* [online]. [cit. 11. 12. 2012]. Dostupné z URL: <http://physics.ujep.cz/~jskvor/NME/DalsiSkripta/Numerika.pdf>.

[14] HORÁK, David. *Diskrétní transformace* [online]. 2012. [cit. 11. 12. 2012]. Dostupné z URL: <http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/diskretni_transformace.pdf>.

[15] KUČERA, Michal. *Segmentovaná vlnková transformace obrazu* [online]. Brno, 2010. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Zdeněk Průša. Dostupné z URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=27873>.

[16] SMÉKAL, Zdeněk. *Číslicové zpracování signálů.* Skripta, Vysoké učení technické v Brně, Brno, aktualizace 2009, s. 202.

[17] MALLAT, Stéphane. *A Wavelet Tour of Signal Processing.* 2nd edition, Academic Press, 1999. ISBN: 0-12-466606-X.

[18] HRBÁČEK, Radek, Pavel RAJMIC, Vítězslav VESELÝ a Jan ŠPIŘÍK. *Řídké reprezentace signálů: úvod do problematiky* [online]. Elektrorevue: časopis pro elektrotechniku. roč. 2011, s. 10 [cit. 20. 05. 2013]. Dostupné z URL: <http://www.elektrorevue.cz/cz/clanky/zpracovani-signalu/0/ridke-reprezentace-signalu--uvod-do-problematiky/>.

[19] MALLAT, Stéphane G. and Zhifeng ZHANG. *Matching Pursuits With Time-Frequency Dictionaries* [online]. IEEE Transactions on Signal Processing. 1993, vol. 41, no. 12, s. 19 [cit. 20. 05. 2013]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=258082>.

[20] *Matching pursuit. Wikipedia: the free encyclopedia* [online]. 23.03.2013, [cit. 20. 05. 2013]. Dostupné z URL: <http://en.wikipedia.org/wiki/Matching_pursuit>.

[21] PERRINET, Laurent, Manuel SAMUELIDES and Simon THORPE. *Sparse spike coding in an asynchronous feed-forward multi-layer neural network using matching pursuit. Neurocomputing* [online]. 2004, vol. 57, s. 125-134 [cit. 20. 05. 2013]. DOI: 10.1016/j.neucom.2004.01.010. Dostupné z URL: <http://linkinghub.elsevier.com/retrieve/pii/S0925231204000670>.

[22] KARAHANOGLU, Nazim Burak and Hakan ERDOGAN. *A* orthogonal matching pursuit: Best-first search for compressed sensing signal recovery. Digital Signal Processing* [online]. 2012, vol. 22, issue 4, s. 555-568 [cit. 20. 05. 2013]. DOI: 10.1016/j.dsp.2012.03.003. Dostupné z URL: <http://linkinghub.elsevier.com/retrieve/pii/S1051200412000656>.

[23] HORE, Alain and Djemel ZIOU. *Image Quality Metrics: PSNR vs. SSIM* [online]. 2010. [cit. 19. 05. 2013]. ISBN 10.1109/ICPR.2010.579. Dostupné z URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5596999>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

A*OMP A* Orthogonal Matching Pursuit

BP Basis Pursuit

CDD Curvature-Driven Diffusions - zakřivením řízené difuze

DCT Discrete Cosine Transform - diskrétní kosinová transformace

DP dolní propust

DTWT Discrete Time Wavelet Transform - diskrétní waveletová transformace s diskrétním časem

DWT Discrete Wavelet Transform - diskrétní waveletová transformace

FT Fourier Transform - Fourierova transformace

GUI Graphical User Interface - grafické uživatelské rozhraní

HP horní propust

HVS Human Visual System

IRLS Iterative Reweighted Least Squares

LARS Least Angle Regression

MiB Mebibajt (je 1048576 Bajtů)

MP Matching Pursuit

MRA Multiresolution analysis - víceúrovňová analýza

MSE Mean Squared Error - střední kvadratická chyba

OMP Orthogonal Matching Pursuit

OOP Object Oriented Programming - objektově orientované programování

PSNR Peak Signal to Noise Ratio - špičkový poměr signálu k šumu

RAM Random-Access Memory

SAD Sum of Absolute Differences - součet absolutních rozdílů

SEA Successive Elimination Algorithm - algoritmus postupné eliminace

SSIM Structural Similarity - strukturální podobnost

WT Wavelet transform - Waveletová transformace

A_k matice aproximačních koeficientů u waveletové transformace

D slovník funkcí pro algoritmy MP a OMP

D_k matice detailních koeficientů u waveletové transformace

D_{2^m} dilatační operátor

$E(x)$ chyba interpolace

F obraz funkce po waveletové transformaci

H Hilbertův prostor

J hloubka waveletové dekompozice

$L_n(x)$ Lagrangeův interpolační polynom stupně n

$N_n(x)$ Newtonův interpolační polynom stupně n

$P_n(x)$ polynom stupně n obecně

P_m operátor ortogonální projekce

$P(p)$ priorita umístění při inpaintingu

$S(x)$ funkce označující splajn

W ortogonální matice rozměru $n \times n$

a_{mn} aproximační koeficienty u dvourozměrné DTWT

d_{mn} detailní koeficienty u dvourozměrné DTWT

$f(x)$ funkce obecně

$g_{\gamma n}$ funkce zvaná atom

h_n škálovací filtrační koeficienty (WT)

x_i uzlový bod polynomu

Φ zdrojová oblast pro inpainting

Σ značí sumu

Ω cílová oblast pro inpainting

ϕ_{mn} škálová funkce

μ střední luminance

ρ^2 kvadratická odchylka

σ standardní odchylka

ψ_{mn} waveletová funkce

$\oplus$ diskrétní (přímý) součet podprostorů

SEZNAM PŘÍLOH

A ORIGINÁLNÍ TESTOVACÍ OBRAZY

Obr. A.1: Krajina.

Obr. A.2: Luxor.

Obr. A.3: Mraky.

Obr. A.4: Tráva.

Obr. A.5: Geometrické tvary.

Obr. A.6: Lena.

Obr. A.7: Budova.

B PŘÍKLAD OBRAZU PO ZPRACOVÁNÍ JEDNOTLIVÝMI METODAMI

Obr. B.1: Metoda aproximace křivkou.

Obr. B.2: Metoda aproximace plochou.

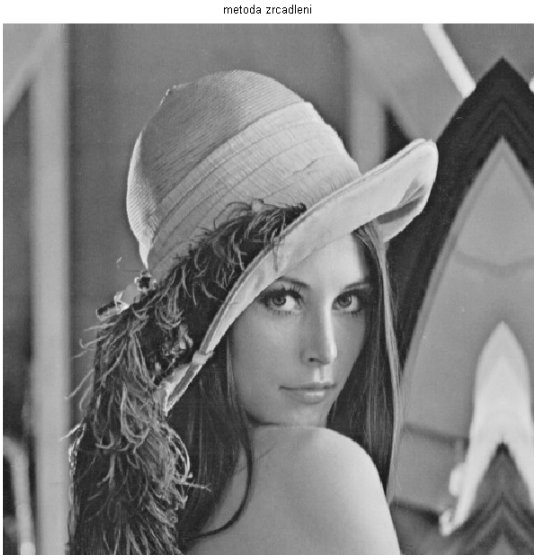
Obr. B.3: Metoda dvojité aproximace.

Obr. B.4: Metoda s detekcí hran.

Obr. B.5: Metoda mediánového filtru.

Obr. B.6: Metoda OMP.

Obr. B.7: Metoda waveletového rozkladu.



Obr. B.8: Metoda zrcadlení.

C TABULKA HODNOT PSNR A SSIM PRO JEDNOTLIVÉ METODY A TYPY OBRAZŮ

Tab. C.1: Výsledky testování metod na různých typech obrazu - část 1

Metoda	Typ obrazu	PSNR [dB]	SSIM [-]
metoda aproximace křivkou	lena	29.12	0.4922
	krajina	30.12	0.5033
	budova	29.49	0.3538
	luxor	30.32	0.5472
	mraky	33.09	0.8661
	trava	27.18	0.0274
	tvary	34.59	0.8932
metoda aproximace plochou	lena	29.09	0.4903
	krajina	29.86	0.5418
	budova	29.01	0.3880
	luxor	30.64	0.6152
	mraky	33.08	0.8971
	trava	27.08	0.0442
	tvary	31.80	0.8004
metoda dvojité aproximace	lena	28.71	0.4377
	krajina	29.24	0.4378
	budova	29.43	0.3519
	luxor	30.40	0.5092
	mraky	31.95	0.7965
	trava	27.64	0.0110
	tvary	34.70	0.8978
metoda mediánového filtru	lena	29.12	0.4981
	krajina	30.10	0.5424
	budova	29.61	0.3759
	luxor	30.23	0.5596
	mraky	33.22	0.8884
	trava	27.16	0.0291
	tvary	34.57	0.8900

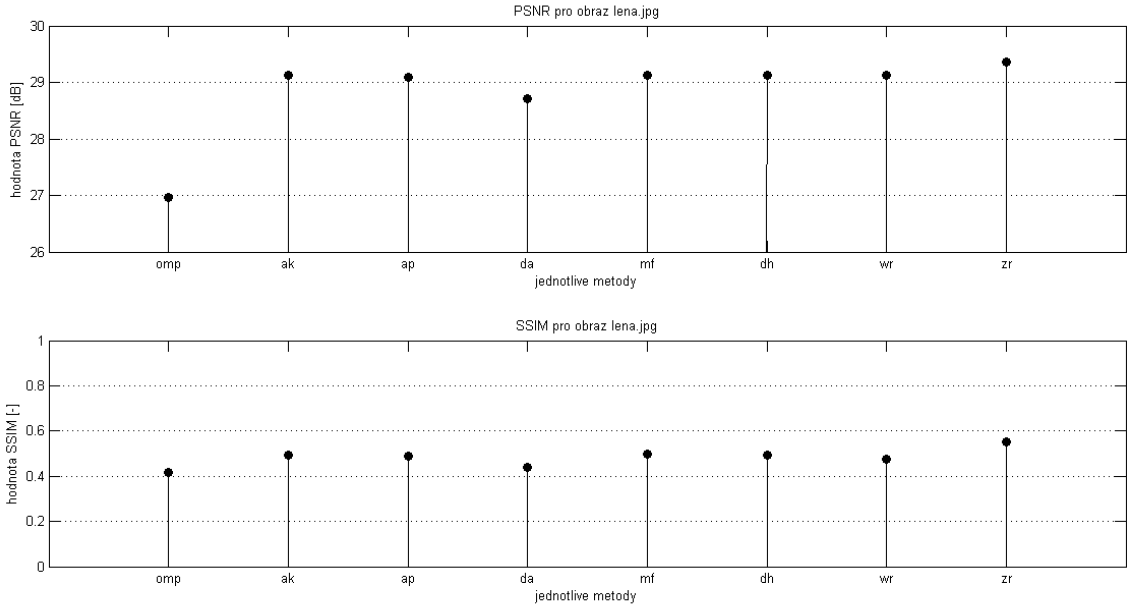
Tab. C.2: Výsledky testování metod na různých typech obrazu - část 2

Metoda	Typ obrazu	PSNR [dB]	SSIM [-]
metoda s detekcí hran	lena	29.12	0.4922
	krajina	30.12	0.5033
	budova	29.49	0.3538
	luxor	30.32	0.5472
	mraky	33.09	0.8661
	trava	27.18	0.0274
	tvary	34.59	0.8932
metoda waveletového rozkladu	lena	29.12	0.4764
	krajina	30.12	0.4987
	budova	29.48	0.3507
	luxor	30.34	0.5442
	mraky	33.06	0.8600
	trava	27.18	0.0245
	tvary	34.60	0.8928
metoda OMP	lena	26.96	0.4147
	krajina	29.48	0.5185
	budova	34.45	0.2178
	luxor	28.45	0.5292
	mraky	33.04	0.8194
	trava	28.12	0.0552
	tvary	31.80	0.7002
metoda zrcadlení	lena	29.36	0.5509
	krajina	30.98	0.5016
	budova	30.42	0.4046
	luxor	32.02	0.5773
	mraky	37.75	0.8638
	trava	27.34	0.0261
	tvary	34.07	0.8830

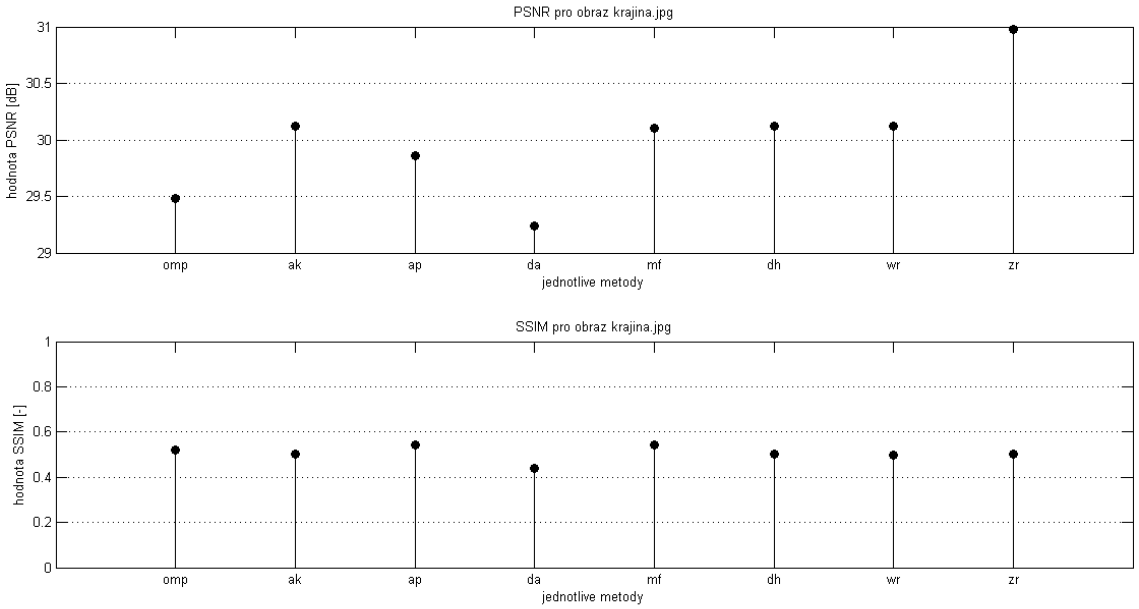
D GRAFICKÉ ZOBRAZENÍ PSNR A SSIM PRO JEDNOTLIVÉ METODY A TYPY OBRAZŮ

Zkratky pro jednotlivé metody, použité při popisu osy x:

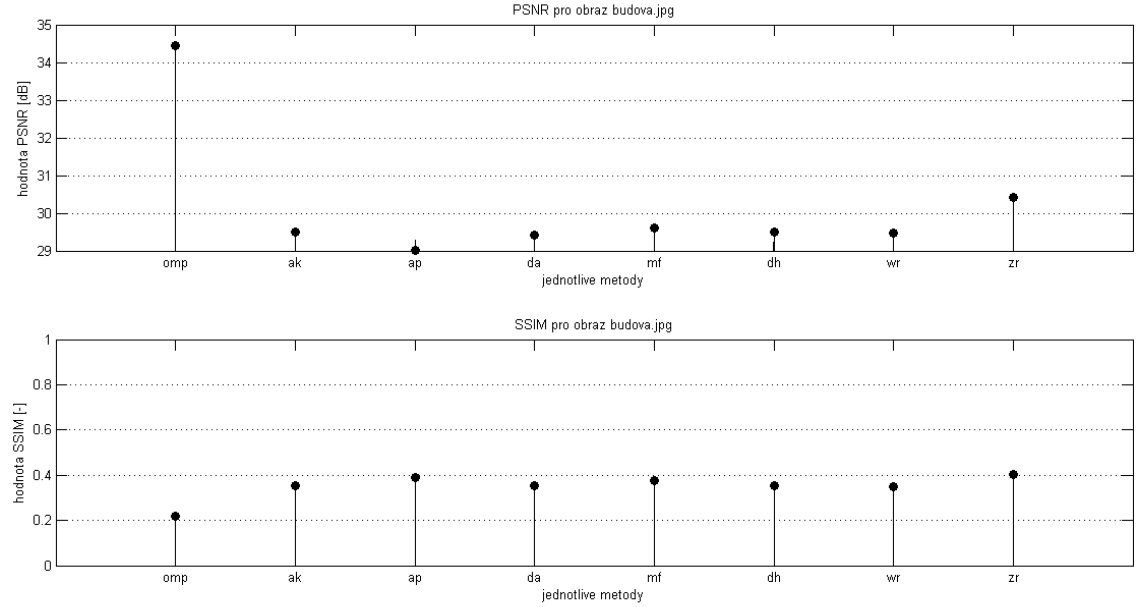
- omp - metoda OMP,
- ak - metoda aproximace křivkou,
- ap - metoda aproximace plochou,
- da - metoda dvojité aproximace,
- mf - metoda mediánového filtru,
- dh - metoda s detekcí hran,
- wr - metoda waveletového rozkladu,
- zr - metoda zrcadlení.



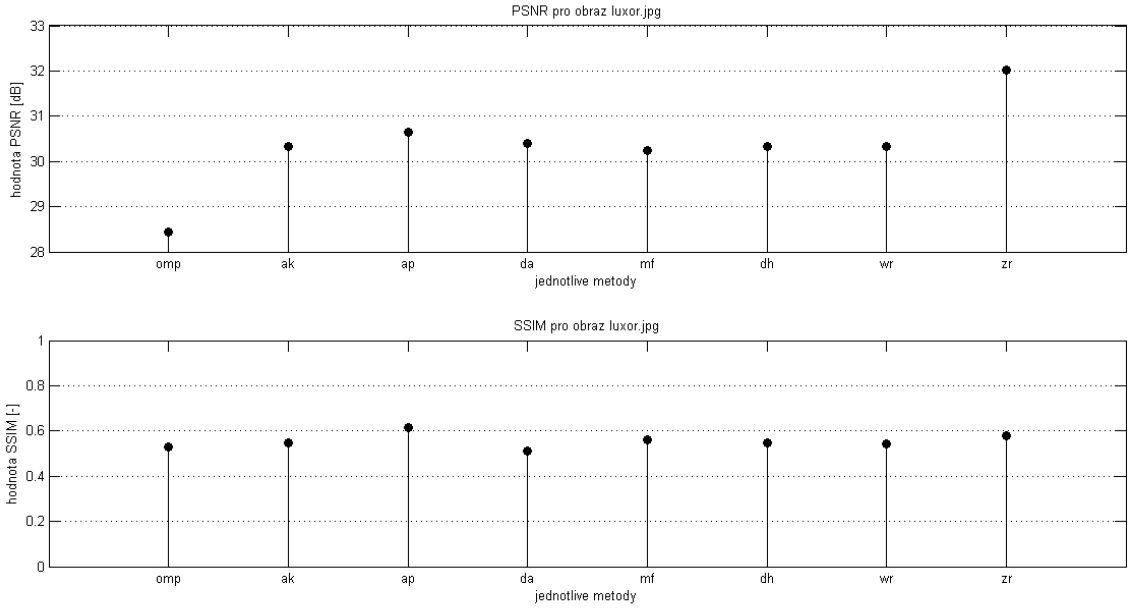
Obr. D.1: Hodnoty PSNR a SSIM pro obraz lena.



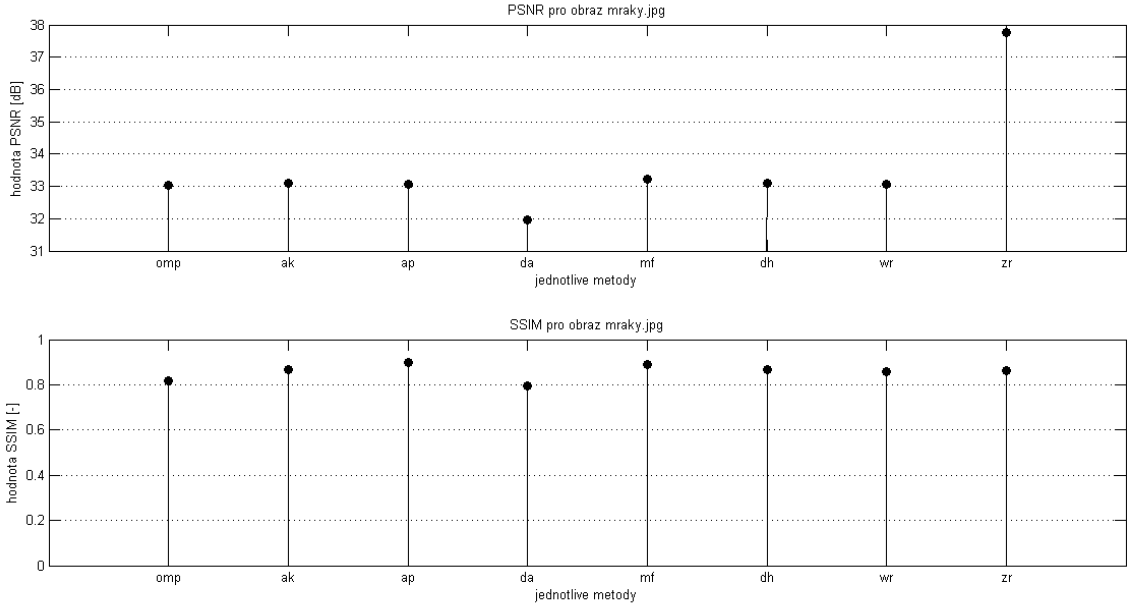
Obr. D.2: Hodnoty PSNR a SSIM pro obraz krajina.



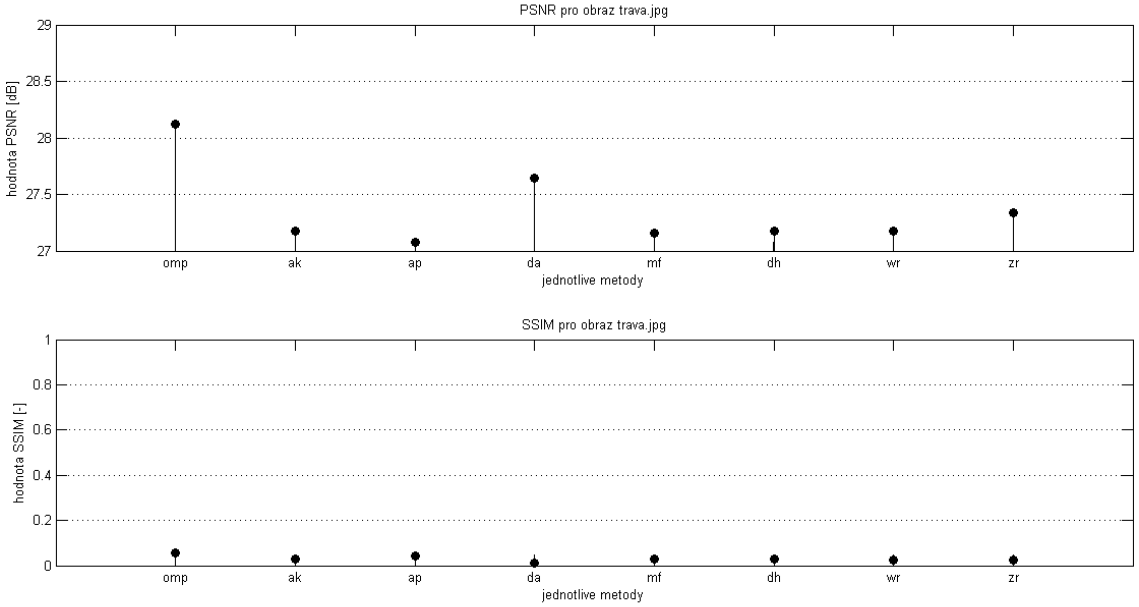
Obr. D.3: Hodnoty PSNR a SSIM pro obraz budova.



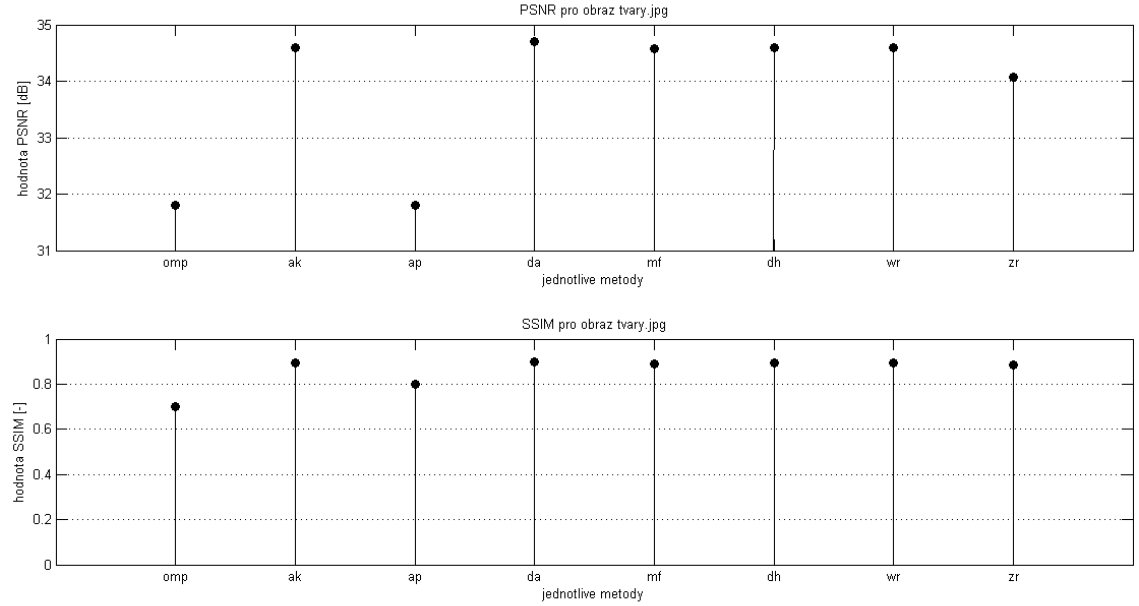
Obr. D.4: Hodnoty PSNR a SSIM pro obraz luxor.



Obr. D.5: Hodnoty PSNR a SSIM pro obraz mraky.



Obr. D.6: Hodnoty PSNR a SSIM pro obraz tráva.



Obr. D.7: Hodnoty PSNR a SSIM pro obraz geometrické tvary.